```
LIBDIR  = ../lib
CC      = gcc
# IPC: NAO PODE USAR -g NO KERNEL !!!
OPT     = -O6
CKFLAGS = -Wall ${OPT} -D_KERNEL -I${LIBDIR}
CNKFLAGS= -Wall -g -I${LIBDIR}

KERNEL  = /kernel
MODUTIL = ../util/modutil
MODPROBE= ../util/modprobe

TAMFILA = -DFILA8K

default:        all

all:            gtasnap gtasnap.conf llc1mux llc1mux.conf       \
                vlan vlan.conf vlt vlt.conf                     \
                plumber checkplumb gtadebug

clean:
                -rm -f gtasnap llc1mux vlt vlan gtadebug
                -rm -f plumber aplic checkplumb
                -rm -f gtasnap.conf llc1mux.conf vlt.conf vlan.conf
                -rm -f *.o core a.out

veryclean:      clean
                -rm -f gtasnap.c llc1mux.c vlt.c vlan.c
                -rm -f go

plumber:        plumber.c plumber.h
                gcc ${CNKFLAGS} -o plumber plumber.c ${LIBDIR}/libgta.a -lsocket

checkplumb:     checkplumb.c plumber.h
                gcc ${CNKFLAGS} -o checkplumb checkplumb.c ${LIBDIR}/libgta.a -l
socket

install:        ${KERNEL}/drv/gtasnap           \
                ${KERNEL}/drv/gtasnap.conf      \
                ${KERNEL}/drv/llc1mux           \
                ${KERNEL}/drv/llc1mux.conf      \
                ${KERNEL}/drv/vlt               \
                ${KERNEL}/drv/vlt.conf          \
                ${KERNEL}/drv/vlan              \
                ${KERNEL}/drv/vlan.conf         \
                ${KERNEL}/strmod/gtadebug

reinstall:      plumberoff      \
                install         \
                plumberon

plumberoff:     plumber
                ./stop

plumberon:      plumber checkplumb
```



```
                ./start `hostname | sed 's/\..*$$//'`

${KERNEL}/drv/llc1mux.conf:     llc1mux.conf
                cp -p $? $@

${KERNEL}/drv/gtasnap.conf:     gtasnap.conf
                cp -p $? $@

${KERNEL}/drv/vlt.conf:         vlt.conf
                cp -p $? $@

${KERNEL}/drv/vlan.conf:        vlan.conf
                cp -p $? $@

${KERNEL}/drv/llc1mux:          llc1mux
                cp -p llc1mux $@
                ${MODUTIL} unload llc1mux

${KERNEL}/drv/gtasnap:          gtasnap
                cp -p gtasnap $@
                ${MODUTIL} unload gtasnap

${KERNEL}/drv/vlt:              vlt
                cp -p vlt $@
                ${MODUTIL} unload vlt

${KERNEL}/drv/vlan:             vlan
                cp -p vlan $@
                ${MODUTIL} unload vlan

${KERNEL}/strmod/gtadebug:      gtadebug
                cp -p gtadebug $@
                ${MODUTIL} unload gtadebug

reload:         plumberoff
                ${MODUTIL} unload llc1mux
                ${MODUTIL} unload gtasnap
                ${MODUTIL} unload vlt
                ${MODUTIL} unload vlan
                ${MODUTIL} unload gtadebug

load:
                ${MODPROBE} /dev/llc1mux
                ${MODPROBE} /dev/gtasnap
                ${MODPROBE} /dev/vlt
                ${MODPROBE} /dev/vlan

llc1mux:        llc1mux.c ../lib/gtallc1.h
                ${CC} ${CKFLAGS} ${TAMFILA} -c llc1mux.c
                ld -r -o llc1mux llc1mux.o
                chmod +x llc1mux
                rm llc1mux.o

gtasnap:        gtasnap.c ../lib/gtallc1.h
```

```
		${CC} ${CKFLAGS} ${TAMFILA} -c gtasnap.c
		ld -r -o gtasnap gtasnap.o
		chmod +x gtasnap
		rm gtasnap.o

vlt:		vlt.c ../lib/gtallc1.h
		${CC} ${CKFLAGS} ${TAMFILA} -c vlt.c
		ld -r -o vlt vlt.o
		chmod +x vlt
		rm vlt.o

vlan:		vlan.c ../lib/gtallc1.h
		${CC} ${CKFLAGS} ${TAMFILA} -c vlan.c
		ld -r -o vlan vlan.o
		chmod +x vlan
		rm vlan.o

gtadebug:	gtadebug.c debug/dlpi.dbg
		${CC} ${CKFLAGS} -c gtadebug.c
		ld -r -o gtadebug gtadebug.o
		chmod +x gtadebug
		rm gtadebug.o

create:		install
	add_drv -m '* 0660 root ether' llc1mux
	add_drv -m '* 0660 root ether' gtasnap
	add_drv -m '* 0660 root ether' vlt
	add_drv -m '* 0660 root ether' vlan
	@echo "Edite manualmente o seu /etc/minor_perm:"
	@echo "clone:llc1mux 0660 root ether"
	@echo "clone:gtasnap 0660 root ether"
	@echo "clone:vlt     0660 root ether"
	@echo "clone:vlan    0660 root ether"

remove:		plumberoff
	-rem_drv llc1mux
	-rem_drv gtasnap
	-rem_drv vlt
	-rem_drv vlan
	@echo "Remova manualmente as seguintes linhas do /etc/minor_perm:"
	@echo "clone:llc1mux 0660 root ether"
	@echo "clone:gtasnap 0660 root ether"
	@echo "clone:vlt     0660 root ether"
	@echo "clone:vlan    0660 root ether"

recreate: remove create

aplic:		aplic.c
		${CC} ${CNKFLAGS} -o aplic aplic.c ../lib/libgta.a -lsocket

go:	go.c
	gcc -Wall -O -o go go.c
	chown 0.0 go
	chmod ug+s go
```



```
llc1mux.c:	enlace.pre
	sed s/##MOD##/llc1mux/g < enlace.pre > llc1mux.c

gtasnap.c:	enlace.pre
	sed s/##MOD##/gtasnap/g < enlace.pre > gtasnap.c

vlt.c:		enlace.pre
	sed s/##MOD##/vlt/g < enlace.pre > vlt.c

vlan.c:		enlace.pre
	sed s/##MOD##/vlan/g < enlace.pre > vlan.c

llc1mux.conf:	Makefile
	echo 'name="llc1mux" parent="pseudo" instance=0;' > llc1mux.conf

gtasnap.conf:	Makefile
	echo 'name="gtasnap" parent="pseudo" instance=0;' > gtasnap.conf

vlt.conf:	Makefile
	echo 'name="vlt" parent="pseudo" instance=0;' > vlt.conf

vlan.conf:	Makefile
	echo 'name="vlan" parent="pseudo" instance=0;' > vlan.conf
```

```
/***********************************************************************\

Driver de enlace para STREAMS/ethernet.

Este módulo é uma base para se gerar quatro drivers diferentes:
  llc1mux - IEEE 802.2 Driver
  gtasnap - SNAP sobre 802.2 Driver
  vlt     - IEEE 802.1p/Q vlan tagging - Camada de seleção de VLAN
  vlan    - IEEE 802.1p/Q vlan tagging - Camada de seleção de TypeSpec

***********************************************************************

 Assumindo que estou num ambiente controlado e imutável.

 - Não vou implementar suporte a XID e TEST.
 - Assumindo driver le(4) e/ou llc1mux abaixo.
 - Usar interface DLPI, e não uma interface minha

Regras de projeto:

2) Usar apenas I_PLINK, retornar erro em caso de I_LINK.
3) Não se preocupar com modo promíscuo
4) Não permitir mais de um bind num único SAP

Estrutura de filas:  Filas na entrada e na saída.
  As filas na entrada são para permitir o controle de fluxo dos
  módulos que enviam dados para estas.

Rotinas da interface:

uwput - Recebe dados dos usuários (entrada da descida)
uwsvc - Fila de entrada da descida
ursvc - Fila de saida da subida (buffer)

lrput - Recebe dados dos drivers (entrada da subida)
lwsvc - Fila de saida da descida (buffer)

Estes são desnecessarios...
lwput - Interface interna de descida
urput - Interface interna de subida
lrsvc - Fila de entrada da subida

  $Id: enlace.pre,v 1.3 1999/03/19 17:29:57 jonny Exp $

\***********************************************************************/
```



```
/***********************************************************************\
|*  Dependencias de compilação                                          *|
\***********************************************************************/

#include <sys/types.h>
#include <sys/conf.h>
#include <sys/stropts.h>
#include <sys/stream.h>
#include <sys/kmem.h>
#include <sys/stat.h>
#include <sys/modctl.h>
#include <sys/errno.h>
#include <sys/strsun.h>         //  Não documentado ???

#include <sys/param.h>
#include <sys/ethernet.h>
#include <sys/dlpi.h>
#include <sys/cmn_err.h>
  #include <inet/common.h>
  #include <inet/led.h>
#include <inet/nd.h>

#include <sys/ddi.h>            //  Must be the last of system includes
#include <sys/sunddi.h>

#include "llc.h"
#include "dlpi.h"
#include "gtallc1.h"
#include "vlan.h"


/***********************************************************************\
|*  Definições locais                                                   *|
\***********************************************************************/

#define MULTITHREAD_SAFE                0

#define DEBUG                           0
#define OPTIMIZE_ALLOCB                 1
#define FULL_ALLOCB_COMPATIBILITY       1
#define DIAGNOSTIC                      DEBUG
#define FORCE_MTU                       0       //  Lance Ethernet não suporta

#if 1
#define __INLINE__      inline
#define __PRIVATE__     static
#else
#define __INLINE__
#define __PRIVATE__
#endif

#define PUTNEXT         putnext         //  Poderia ser um putq(q->next) ?

//  Cuidado com estes valores, eles influenciam o tamanho das estruturas
```

```
#define MAX_CLIENTS              (20)     //  Usuarios de um lnode
#define MAX_LNODES               (4)      //  Total de lnodes
#define MAX_UNODES               (MAX_LNODES*MAX_CLIENTS)

#if DEBUG
#define DEBUG_MSG_0(Msg)         cmn_err(CE_NOTE,Msg)
#define DEBUG_MSG_1(Msg,a)       cmn_err(CE_NOTE,Msg,a)
#define DEBUG_MSG_2(Msg,a,b)     cmn_err(CE_NOTE,Msg,a,b)
#define DEBUG_MSG_3(Msg,a,b,c)   cmn_err(CE_NOTE,Msg,a,b,c)
#define Debugger()               asm( "t 0x7d            ! KADB TRAP" )
#else
#define DEBUG_MSG_0(Msg)
#define DEBUG_MSG_1(Msg,a)
#define DEBUG_MSG_2(Msg,a,b)
#define DEBUG_MSG_3(Msg,a,b,c)
#define Debugger()
#endif


/*********************************************************************\
|*  Configuração dependente do módulo objetivo                       *|
\*********************************************************************/

#define __iam##MOD##__
#define isLLC  defined(__iamllc1mux__)
#define isSNAP defined(__iamgtasnap__)
#define isVLT  defined(__iamvlt__)
#define isVLAN defined(__iamvlan__)

#if   isLLC
#define         PCI_SIZE        (3)
#define         MAX_MSG         (1497)
#elif isSNAP
#define         PCI_SIZE        (5)
#define         MAX_MSG         (1492)
#elif isVLT
#define         PCI_SIZE        (2)
#if             FORCE_MTU
#define         MAX_MSG         (1502)
#else
#define         MAX_MSG         (1498)
#endif
#elif isVLAN
#define         PCI_SIZE        (2)
#if             FORCE_MTU
#define         MAX_MSG         (1500)
#else
#define         MAX_MSG         (1496)
#endif
#else
#error "Erro interno, verifique os fontes"
#endif
```



```
/*********************************************************************\
|*  Module Configuration                                             *|
\*********************************************************************/

__PRIVATE__ char modname[] = "##MOD##";

__PRIVATE__ struct module_info ##MOD##_info = {
  0x1232,
  modname,
  0,            //  min msgsize
  MAX_MSG,      //  max msgsize
#if defined(FILA64K)
  65536,        //  hiwater
  8192          //  lowater
#else
#if defined(FILA2K)
  2048,         //  hiwater
  512           //  lowater
#else
#if defined(FILA8K)
  8192,         //  hiwater
  1024          //  lowater
#else
#error "Escolha um tamanho de fila para o controle de fluxo..."
#endif
#endif
#endif
};

/*  Module structures  */

__PRIVATE__ int
        ##MOD##_open( queue_t *, dev_t *, int, int, cred_t * ),
        ##MOD##_close( queue_t *, int, cred_t * ),
        ##MOD##_uwput( queue_t *, mblk_t * ),
        ##MOD##_ursvc( queue_t * ),
        ##MOD##_uwsvc( queue_t * ),
        ##MOD##_lwsvc( queue_t * ),
        ##MOD##_lrput( queue_t *, mblk_t * );

__PRIVATE__ int
        ##MOD##_getinfo( dev_info_t *, ddi_info_cmd_t, void *, void ** ),
        ##MOD##_identify( dev_info_t * ),
        ##MOD##_attach( dev_info_t *, ddi_attach_cmd_t ),
        ##MOD##_detach( dev_info_t *, ddi_detach_cmd_t );

__PRIVATE__ struct qinit urinit = {
  NULL,                 ##MOD##_ursvc,
  ##MOD##_open,         ##MOD##_close,
  NULL,                &##MOD##_info,
  NULL
};

__PRIVATE__ struct qinit uwinit = {
```

```
  ##MOD##_uwput,          ##MOD##_uwsvc,
  NULL,                   NULL,
  NULL,                  &##MOD##_info,
  NULL
};

__PRIVATE__ struct qinit lrinit = {
  ##MOD##_lrput,          NULL,
  NULL,                   NULL,
  NULL,                  &##MOD##_info,
  NULL
};

__PRIVATE__ struct qinit lwinit = {
  NULL,                   ##MOD##_lwsvc,
  NULL,                   NULL,
  NULL,                  &##MOD##_info,
  NULL
};

__PRIVATE__ struct streamtab ##MOD##_tab = { &urinit, &uwinit, &lrinit, &lwinit
};

__PRIVATE__ struct cb_ops ##MOD##_cb_ops = {
  NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
  &##MOD##_tab,
#if MULTITHREAD_SAFE
  D_MP
#else
  0
#endif
};

__PRIVATE__ struct dev_ops ##MOD##_ops = {
  DEVO_REV,
  0,
  ##MOD##_getinfo,
  ##MOD##_identify,
  nulldev,                //  probe
  ##MOD##_attach,
  ##MOD##_detach,
  nodev,                  //  reset
  &##MOD##_cb_ops,
  NULL
};

//  Isso aqui é extern mesmo.  Parece que é colocado pelo modload !!!
extern struct mod_ops mod_driverops;

__PRIVATE__ struct modldrv modldrv = {
  &mod_driverops,
#if   isLLC
  "GTA: LLC1 Multiplexer Driver",
#elif isSNAP
```



```
  "GTA: SNAP Multiplexer Driver",
#elif isVLT
  "GTA: 802.1p/Q VLAN - Tagger",
#elif isVLAN
  "GTA: 802.1p/Q VLAN - EtherSpec",
#else
#error "Erro interno, verifique os fontes"
#endif
  &##MOD##_ops
};

__PRIVATE__ struct modlinkage modlinkage = { MODREV_1, { (void*)&modldrv, NULL }
 };
```

```
/*******************************************************************\
|*  Module private data                                            *|
\*******************************************************************/

typedef struct ether_addr ether_addr_t;

typedef struct {
  int            Size;
  ether_addr_t  *MyAddr;        //  Endereço MAC local
  dl_u           Ack;
} iac_t;

/*  Estrutura para definir cada um dos nodes                       */

typedef struct lnode lnode_t;
typedef struct unode unode_t;

struct unode {
  queue_t       *rdq;           /*  Backref to the unode read queue     */
  queue_t       *wrq;           /*  Backref to the unode write queue    */
  lnode_t       *lnode;         /*  Lower Node where it's bound         */
  ulong          uState;        /*  Maquina de estados                  */
  ulong          uHang;         /*  unode entrou em estado de HANGUP    */
  ulong          uSap;          /*  LSAP associado                      */
};

typedef struct {
  unode_t        *unode;        /*  Ponteiro para o unode cliente       */
  ulong           SAP;          /*  Cache para o SAP associado...       */
} client_t;

struct lnode {
  queue_t       *rdq;           /*  Backref to the lnode read queue     */
  queue_t       *wrq;           /*  Backref to the lnode write queue    */
  iac_t         *iacCache;      /*  Cache para o DL_INFO_ACK            */
  int            RefCount;      /*  Quantos unodes associados ?         */
  mblk_t        *mpCache;
  queue_t       *qCache;
  client_t       clients[MAX_CLIENTS];  //  Lista de unodes associados
};

__PRIVATE__ unode_t unodelist[ MAX_UNODES ];
__PRIVATE__ lnode_t lnodelist[ MAX_LNODES ];

__PRIVATE__ struct {
  ether_addr_t DefaultEtherAddr;
  dev_info_t *dip;
} PrivateData;

__PRIVATE__ ether_addr_t EtherBCast = {{ 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF }};
```



```
/*******************************************************************\
|*  Module private funtions                                        *|
\*******************************************************************/

static inline void
memcpy( void *To, void *From, size_t N ) {
  bcopy( From, To, N );
}

#if DIAGNOSTIC

#define freemsg(a)      \
                if (a) { freemsg(a); a = NULL; } \
                else cmn_err( CE_PANIC, "freemsg invalido: %d", __LINE__ )

#endif

/*******************************************************************/

__PRIVATE__ int
AddClient( lnode_t *lnode, unode_t *unode )
{
  int i;
  client_t *cp;

  cp = &lnode->clients[0];
  for ( i = 0 ; i < MAX_CLIENTS ; ++i ) {
    if ( cp->unode == NULL ) {
      cp->unode = unode;
      cp->SAP = SAP_UNBOUND;
      return 0;
    }
    ++cp;
  }
  return -1;
}

__PRIVATE__ int
ChgClient( lnode_t *lnode, unode_t *unode, ulong SAP )
{
  int i;
  client_t *cp;

  cp = &lnode->clients[0];
  for ( i = 0 ; i < MAX_CLIENTS ; ++i ) {
    if ( cp->unode == unode ) {
      cp->SAP = SAP;
      return 0;
    }
    ++cp;
  }
  return -1;
}
```

```
__PRIVATE__ unode_t *
FindClient( lnode_t *lnode, ulong SAP )
{
  int i;
  client_t *cp;

  cp = &lnode->clients[0];
  for ( i = 0 ; i < MAX_CLIENTS ; ++i ) {
    if ( cp->SAP == SAP ) {
      return cp->unode;
    }
    ++cp;
  }
  return NULL;
}

__PRIVATE__ int
DelClient( lnode_t *lnode, unode_t *unode )
{
  int i;
  client_t *cp;

  cp = &lnode->clients[0];
  for ( i = 0 ; i < MAX_CLIENTS ; ++i ) {
    if ( cp->unode == unode ) {
      cp->unode = NULL;
      return 0;
    }
    ++cp;
  }
  return -1;
}

/***********************************************************************/

__PRIVATE__ lnode_t *
GetNewLNode( queue_t *wrq )
{
  int i;

  /* beginmutex */
  for ( i = 0 ; i < MAX_LNODES ; ++i ) {
    if ( lnodelist[ i ].wrq == NULL ) {
      DEBUG_MSG_1( "GetNewLNode: Achei nodo livre na posicao %d.", i );
      lnodelist[ i ].wrq = wrq;
      /* endmutex */
      return &lnodelist[ i ];
    }
  }
  /* endmutex */
  return NULL;
}

__PRIVATE__ lnode_t *
```



```
FindLNode( queue_t *wrq )
{
  int i;

  for ( i = 0 ; i < MAX_LNODES ; ++i ) {
    if ( lnodelist[ i ].wrq == wrq ) {
      DEBUG_MSG_1( "FindLNode: Achei nodo na posicao %d.", i );
      return &lnodelist[ i ];
    }
  }
  return NULL;
}

/***********************************************************************/

__PRIVATE__ __INLINE__ void
DoUnbind( unode_t *unode )
{
  /*  begin mutex  */
  ChgClient( unode->lnode, unode, SAP_UNBOUND );
  DEBUG_MSG_1( "##MOD##: Unbound from 0x%04X", unode->uSap );
  unode->uSap = SAP_UNBOUND;
  unode->uState = DL_UNBOUND;
  /*  end mutex  */
}

/***********************************************************************/

__PRIVATE__ void
DoDetach( unode_t *unode )
{
  lnode_t *lnode;

  lnode = unode->lnode;
  DEBUG_MSG_1( "##MOD##: Detached from ppa #%d", lnode - lnodelist );
  /* enter mutex */
  lnode->RefCount--;
  unode->uState = DL_UNATTACHED;
  DelClient( lnode, unode );
  unode->lnode = NULL;
  /* leave mutex */
}

/***********************************************************************/

__PRIVATE__ __INLINE__ void
DoHangup( unode_t *unode )
{
  /*  begin mutex  */
  unode->uSap   = SAP_UNBOUND;
  unode->uState = DL_UNATTACHED;
  unode->lnode  = NULL;
  unode->uHang  = 1;
  /* leave mutex */
```

```
}

/**************************************************************************/

#if 0
__PRIVATE__ void
DumpMsg( mblk_t *mp )
{
  char Aux1[ 80 ], Aux2[ 80 ];
  const char *p;
  int n;

  if ( mp == NULL ) {
    cmn_err( CE_WARN, "DumpMsg: Trying to dump a NULL pointer" );
    return;
  }

  n = MBLKL(mp);
  cmn_err( CE_CONT, "Message Block Dump - Type: " );
  switch ( DB_TYPE(mp) ) {
    case M_PCPROTO:
      cmn_err( CE_CONT, "M_PCPROTO" );
      break;
    case M_PROTO:
      cmn_err( CE_CONT, "M_PROTO" );
      break;
    case M_IOCTL:
      cmn_err( CE_CONT, "M_IOCTL" );
      break;
    default:
      cmn_err( CE_CONT, "%02X", DB_TYPE(mp) );
      break;
  }
  cmn_err( CE_CONT, ", Size: %3d\n", n );

  sprintf( Aux2, "Data:" );
  p = mp->b_rptr;
  if ( n > 16 ) n = 16;
  while ( n-- > 0 ) {
    strcat( Aux2, sprintf( Aux1, " %02X", *p++ & 0xFF ) );
  }
  strcat( Aux2, "\n" );
  cmn_err( CE_CONT, Aux2 );
}
#endif

/**************************************************************************/

#if OPTIMIZE_ALLOCB
__PRIVATE__ mblk_t *
ReallocMsgBlock( mblk_t *mp, int Size, uint Pri )
{
  /*  Tenta reaproveitar os blocos pre-alocados.                          */
  /*  Deve ser equivalente a um freemsg seguido de um allocb              */
```



```
  /*  Obs: argumento recebido é uma msg, argumento devolvido é blk        */

  static ulong Hits = 0, Misses = 0;

  int OldSize = mp->b_datap->db_lim - mp->b_datap->db_base;

  if ( OldSize >= Size && mp->b_datap->db_ref == 1 ) {
    /* Reaproveita.                                                       */
    ++Hits;
#if DEBUG > 2
    cmn_err( CE_NOTE, "MsgCacheHit: Size: %d, Usage: %d, New: %d, Hits: %d/%d",
      OldSize,
      MBLKL(mp),
      Size,
      Hits,
      Hits + Misses
    );
#endif

    mp->b_wptr = mp->b_rptr = mp->b_datap->db_base;
    #if FULL_ALLOCB_COMPATIBILITY
    DB_TYPE(mp) = M_DATA;
    #endif
    if ( mp->b_cont != NULL ) {
      freemsg( mp->b_cont );
      mp->b_cont = NULL;
    }
    return mp;
  }
  else {
    ++Misses;
#if DEBUG > 2
    cmn_err( CE_NOTE, "MsgCacheMiss: Size: %d, Usage: %d, New: %d, Hits: %d/%d",
      OldSize,
      MBLKL(mp),
      Size,
      Hits,
      Hits + Misses
    );
#endif

    freemsg( mp );
    return allocb( Size, Pri );
  }
}
#else
__PRIVATE__ inline mblk_t *
ReallocMsgBlock( mblk_t *mp, int Size, uint Pri )
{
  freemsg( mp );
  return allocb( Size, Pri );
}
#endif
```

```
/*********************************************************************\
|  Rotinas de conversão de prioridade.  Não serão utilizadas as       |
|  prioridades 1 e 2 do 802.1p, sendo estas mapeadas na prioridade zero. |
\*********************************************************************/

inline u8
PriVlanToStreams( u8 vid )
{
  if ( vid <= 2 ) {
    return 0;
  }
  if ( vid >= 8 ) {
    return 7 * 8;
  }
  return vid * 8;
}

inline u8
PriStreamsToVlan( u8 band )
{
  return band >= 24 ? band / 8 : 0;
}
```



```
/*********************************************************************\
|*  Funcoes de tratamento de mensagens                               *|
\*********************************************************************/

__PRIVATE__ void
DevolveErrorAck( queue_t *q, mblk_t *mp, ulong Prim, ulong Error, ulong UnixErro
r )
{
  dl_error_ack_t *eap;

  mp = ReallocMsgBlock( mp, sizeof (dl_error_ack_t), BPRI_MED );
  if ( mp == NULL ) {
    cmn_err( CE_WARN, "##MOD##.DevolveErrorAck: No memory..." );
    return;
  }
  DB_TYPE(mp) = M_PCPROTO;
  mp->b_wptr += sizeof (dl_error_ack_t);
  eap = (dl_error_ack_t*)mp->b_rptr;
  eap->dl_primitive = DL_ERROR_ACK;
  eap->dl_error_primitive = Prim;
  eap->dl_errno = Error;
  eap->dl_unix_errno = UnixError;
  qreply( q, mp );
}

/*********************************************************************/

__PRIVATE__ void
DevolveUderrorInd( queue_t *q, mblk_t *mp, ulong Error, ulong UnixError )
{
  mblk_t *tmp;
#if   isLLC
  llc1_uderror_ind_t  *ude;
  llc1_unitdata_req_t *udr;
#elif isSNAP
  le_uderror_ind_t    *ude;
  le_unitdata_req_t   *udr;
#elif isVLT
  le_uderror_ind_t    *ude;
  le_unitdata_req_t   *udr;
#elif isVLAN
  le_uderror_ind_t    *ude;
  le_unitdata_req_t   *udr;
#else
#error "Erro interno, verifique os fontes"
#endif

  tmp = allocb( sizeof (*ude), BPRI_MED );
  if ( tmp == NULL ) {
    cmn_err( CE_WARN, "##MOD##.DevolveUderrorInd: No memory..." );
    freemsg( mp );
    return;
  }
  DB_TYPE(tmp) = M_PCPROTO;
```

```
  tmp->b_wptr += sizeof (*ude);
  ude = (void *)tmp->b_rptr;
  ude->dl.dl_primitive = DL_UDERROR_IND;
  ude->dl.dl_dest_addr_length = sizeof(ude->addr);
  ude->dl.dl_dest_addr_offset = OFFSET( ude, &(ude->addr) );
  ude->dl.dl_unix_errno = UnixError;
  ude->dl.dl_errno = Error;
  udr = (void*)mp->b_rptr;
  memcpy( &ude->addr, &udr->dst, sizeof (ude->addr) );
  qreply( q, tmp );
  freemsg( mp );
}

/*************************************************************************/

__PRIVATE__ void
DevolveOkAck( queue_t *q, mblk_t *mp, ulong Prim )
{
  dl_ok_ack_t *oap;

  mp = ReallocMsgBlock( mp, sizeof (dl_ok_ack_t), BPRI_MED );
  if ( mp == NULL ) {
    cmn_err( CE_WARN, "##MOD##.DevolveOkAck: No memory..." );
    return;
  }
  DB_TYPE(mp) = M_PCPROTO;
  mp->b_wptr += sizeof (dl_ok_ack_t);
  oap = (dl_ok_ack_t*)mp->b_rptr;
  oap->dl_primitive = DL_OK_ACK;
  oap->dl_correct_primitive = Prim;
  qreply( q, mp );
}

/*************************************************************************/

__PRIVATE__ __INLINE__ void
DevolveIoctl( queue_t * q, mblk_t *mp, u8 type, int error, int rval )
{
  struct iocblk *ioc;

  ioc = (struct iocblk *)mp->b_rptr;
  DB_TYPE(mp) = type;                   //  M_IOCACK || M_IOCNAK
  ioc->ioc_error = error;               //  Error Code  (errno ?)
  ioc->ioc_rval = rval;                 //  Return Value
  qreply( q, mp );
}

/*************************************************************************/

__PRIVATE__ void
DevolveErrorMsg( queue_t *q, mblk_t *mp, u8 RdError, u8 WrError )
{
  mblk_t *tmp;
```



```
  tmp = ReallocMsgBlock( mp, 2, BPRI_HI );

  if ( tmp == NULL ) {
    cmn_err( CE_WARN, "##MOD##.DevolveErrorMsg: No memory..." );
    return;
  }
  DB_TYPE(mp) = M_ERROR;
  *mp->b_wptr++ = RdError;
  *mp->b_wptr++ = WrError;
  qreply( q, mp );
}

/*************************************************************************/

__PRIVATE__ __INLINE__ void
EnviaInfoReq( queue_t *q )
{
  mblk_t *mp;

  if ( ( mp = allocb( sizeof (dl_info_req_t), BPRI_MED ) ) == NULL ) {
    cmn_err( CE_WARN, "##MOD##.EnviaInfoReq: No memory..." );
    return;
    //  XXX  Deveria esperar algum tempo para pedir de novo, vou entrar
    //       em deadlock.
  }
  DB_TYPE(mp) = M_PCPROTO;
  mp->b_wptr += sizeof (dl_info_req_t);
  ((dl_info_req_t*)mp->b_rptr)->dl_primitive = DL_INFO_REQ;
  putnext( q, mp );
}

/*************************************************************************/

static inline void
CheckSize( queue_t *wq, mblk_t *mp, int Size, int Prim )
{
  if ( MBLKL(mp) < Size ) {
    cmn_err( CE_WARN, "##MOD##: Primitiva invalida, tamanho %d", MBLKL(mp) );
    DevolveErrorAck( wq, mp, Prim, DL_BADPRIM, 0 );
    return;
  }
}
```

```
/**********************************************************************/
/*  Module Processing Funtions                                        */
/**********************************************************************/

__PRIVATE__ __INLINE__ void
ProcessaInfoAck( lnode_t *lnode, mblk_t *mp )
{
  iac_t *ia;
  int i;

  if ( lnode->iacCache != NULL ) {
    cmn_err( CE_WARN,
             "##MOD##.ProcessaInfoAck: Already hold a copy of INFO_ACK" );
    goto Error;
  }

  if ( lnode->qCache == NULL || lnode->mpCache == NULL ) {
    cmn_err( CE_WARN, "##MOD##:ProcessaInfoAck: Invalid lnode State !" );
  Error:
    freemsg( mp );
    return;
  }

  i = MBLKL(mp) +                               //  Tamanho dos dados
      ( (char *)&(ia->Ack) - (char *)ia );      //  Área inicial do iac_t

  if ( ( ia = kmem_alloc( i, KM_NOSLEEP ) ) == NULL ) {
    cmn_err( CE_WARN, "##MOD##.ProcessaInfoAck: No memory" );
    freemsg( mp );
    DEBUG_MSG_2( "Unable to link lnode 0x%08x, request from 0x%08X",
                                                 lnode, lnode->qCache );
    DevolveIoctl( lnode->qCache, lnode->mpCache, M_IOCNAK, EAGAIN, -1 );
    lnode->qCache = NULL;
    lnode->mpCache = NULL;
    return;
  }

  ia->Size = i;
  memcpy( &ia->Ack, mp->b_rptr, MBLKL(mp) );
  freemsg( mp );

  lnode->iacCache = ia;
  lnode->iacCache->MyAddr =
      (ether_addr_t*)&ia->Ack.data[ ia->Ack.dl.info_ack.dl_addr_offset ];

  /*  Avisar o unode master que completou o link                      */
  DEBUG_MSG_2( "Linking lnode 0x%08x, request from 0x%08X", lnode,
                                                     lnode->qCache );
  DevolveIoctl( lnode->qCache, lnode->mpCache, M_IOCACK, 0, 0 );
  lnode->qCache = NULL;
  lnode->mpCache = NULL;
}

/**********************************************************************/
```



```
__PRIVATE__ __INLINE__ void
ProcessaBindReq( queue_t *q, mblk_t *mp )
{
  dl_bind_req_t *brp;
  unode_t *unode;
  u_long Sap;
#if   isLLC
  llc1_bind_ack_t *bap;
#elif isSNAP
  le_bind_ack_t   *bap;
#elif isVLT
  le_bind_ack_t   *bap;
#elif isVLAN
  le_bind_ack_t   *bap;
#else
#error "Erro interno, verifique os fontes"
#endif

  unode = (unode_t *)q->q_ptr;

  if ( unode->uState != DL_UNBOUND ) {
    DevolveErrorAck( q, mp, DL_BIND_REQ, DL_OUTSTATE, 0 );
    return;
  }

  brp = (dl_bind_req_t *)mp->b_rptr;
  if ( brp->dl_service_mode != DL_CLDLS ) {
    DevolveErrorAck( q, mp, DL_BIND_REQ, DL_UNSUPPORTED, 0 );
    return;
  }

  switch ( brp->dl_xidtest_flg & (DL_AUTO_XID|DL_AUTO_TEST) ) {
    case DL_AUTO_XID:
      DevolveErrorAck( q, mp, DL_BIND_REQ, DL_NOXIDAUTO, 0 );
      return;

    case DL_AUTO_TEST:
      DevolveErrorAck( q, mp, DL_BIND_REQ, DL_NOTESTAUTO, 0 );
      return;

    case DL_AUTO_XID|DL_AUTO_TEST:
      DevolveErrorAck( q, mp, DL_BIND_REQ, DL_NOAUTO, 0 );
      return;
  }

  Sap = brp->dl_sap;

  //  Verifica range do Sap desejado.
#if   isLLC
  if ( ( Sap > 0xFF ) || ( Sap & 0x01 ) ) {
#elif isSNAP
  if ( Sap > 0xFFFF ) {
#elif isVLT
```

```
  if ( Sap > 0x0FFF ) {
#elif isVLAN
  if ( Sap > 0xFFFF ) {
#else
#error "Erro interno, verifique os fontes"
#endif
    DEBUG_MSG_1( "##MOD##.ProcessaBindReq: BADADDR, addr = 0x%02X", Sap );
    DevolveErrorAck( q, mp, DL_BIND_REQ, DL_BADADDR, 0 );
    return;
  }

  if ( FindClient( unode->lnode, Sap ) != NULL ) {
    // SAP requerido já está em uso por outro unode
    DEBUG_MSG_1( "##MOD##.ProcessaBindReq: BUSY, addr = 0x%02X", Sap );
    DevolveErrorAck( q, mp, DL_BIND_REQ, DL_SYSERR, EBUSY );
    return;
  }

  if ( ChgClient( unode->lnode, unode, Sap ) != 0 ) {
    DEBUG_MSG_1( "##MOD##.ProcessaBindReq: lnode client unavailable, addr = 0x%0
2X", Sap );
    DevolveErrorAck( q, mp, DL_BIND_REQ, DL_SYSERR, ENOSR );
    return;
  }
  unode->uSap = Sap;
  DEBUG_MSG_1( "##MOD##: Bound to 0x%04X", Sap );

  /*  Como sou DL_CLDLS, não trato dl_max_conind nem dl_conn_mgmt       */

  if ( ( mp = ReallocMsgBlock( mp,
                               sizeof (*bap),
                               BPRI_MED
              ) ) == NULL ){
    cmn_err( CE_WARN, "##MOD##.ProcessaBindReq: No memory" );
    return;
  }
  DB_TYPE(mp) = M_PCPROTO;
  mp->b_wptr += sizeof (*bap);
  bap = (void *)mp->b_rptr;
  bap->dl.dl_primitive = DL_BIND_ACK;
  bap->dl.dl_sap = Sap;
  bap->dl.dl_addr_length = sizeof (bap->addr);
  bap->dl.dl_addr_offset = OFFSET( bap, &(bap->addr) );
  bap->dl.dl_max_conind = 0;
  bap->dl.dl_xidtest_flg = 0;
  ether_copy( unode->lnode->iacCache->MyAddr, &bap->addr.ether );
  bap->addr.sap = Sap;

  unode->uState = DL_IDLE;
  qreply( q, mp );
}

/************************************************************************/
```



```
__PRIVATE__ __INLINE__ void
ProcessaPhysAddrReq( queue_t *q, mblk_t *mp )
{
  dl_phys_addr_req_t *parp;
  unode_t *unode;
  lnode_t *lnode;
#if   isLLC
  llc1_phys_addr_ack_t *paap;
#elif isSNAP
  le_phys_addr_ack_t   *paap;
#elif isVLT
  le_phys_addr_ack_t   *paap;
#elif isVLAN
  le_phys_addr_ack_t   *paap;
#else
#error "Erro interno, verifique os fontes"
#endif

  unode = (unode_t *)q->q_ptr;

  if ( unode->uState == DL_UNATTACHED ) {
    DevolveErrorAck( q, mp, DL_PHYS_ADDR_REQ, DL_OUTSTATE, 0 );
    return;
  }

  parp = (dl_phys_addr_req_t *)mp->b_rptr;

  switch ( parp->dl_addr_type ) {
    case DL_FACT_PHYS_ADDR:
    case DL_CURR_PHYS_ADDR:
      break;

    default:
      DevolveErrorAck( q, mp, DL_PHYS_ADDR_REQ, DL_NOTSUPPORTED, 0 );
      return;
  }

  if ( ( mp = ReallocMsgBlock( mp,
                               sizeof (*paap),
                               BPRI_MED
              ) ) == NULL ){
    cmn_err( CE_WARN, "##MOD##.ProcessaPhysAddrReq: No memory" );
    return;
  }
  DB_TYPE(mp) = M_PCPROTO;
  mp->b_wptr += sizeof (*paap);
  paap = (void *)mp->b_rptr;
  paap->dl.dl_primitive = DL_PHYS_ADDR_ACK;
  paap->dl.dl_addr_length = sizeof (paap->addr.ether);
  paap->dl.dl_addr_offset = OFFSET( paap, &(paap->addr) );

  if ( parp->dl_addr_type == DL_CURR_PHYS_ADDR ) {
    lnode = unode->lnode;
    if ( lnode == NULL || lnode->iacCache == NULL ) {
```

```
      if ( lnode == NULL ) {
        cmn_err( CE_WARN, "##MOD##: DL_PHYS_ADDR_REQ while unattached" );
        cmn_err( CE_WARN, "##MOD##: State = 0x%02X", unode->uState );
      }
      else {
        cmn_err( CE_WARN, "##MOD##: DL_PHYS_ADDR_REQ / iacCache uninitialized");
      }
      ether_copy( &PrivateData.DefaultEtherAddr, &paap->addr.ether );
    }
    else {
      ether_copy( lnode->iacCache->MyAddr, &paap->addr.ether );
    }
  }
  else {
    ether_copy( &PrivateData.DefaultEtherAddr, &paap->addr.ether );
  }
  qreply( q, mp );
}

/*************************************************************************/

__PRIVATE__ __INLINE__ void
ProcessaUnbindReq( queue_t *q, mblk_t *mp )
{
  unode_t *unode;

  unode = (unode_t *)q->q_ptr;

  if ( unode->uState != DL_IDLE ) {
    DevolveErrorAck( q, mp, DL_BIND_REQ, DL_OUTSTATE, 0 );
    return;
  }

  DoUnbind( unode );

  DevolveOkAck( q, mp, DL_BIND_REQ );
}

/*************************************************************************/

__PRIVATE__ __INLINE__ void
ProcessaUnitDataReq( queue_t *q, mblk_t *mp )
{
  //  Basicamente, eu apenas adiciono a PCI e passo adiante, sem modificar.

  unode_t *unode;
  lnode_t *lnode;
  mblk_t *tmp;
  int Size;
  u_long Sap;

#if   isLLC
  le_unitdata_req_t   *ludr;
  llc1_unitdata_req_t *uudr;
```



```
  llc_pci_tipo1_t *pci;
#elif isSNAP
  llc1_unitdata_req_t *ludr;
  le_unitdata_req_t   *uudr;
  snap_pci_t *pci;
#elif isVLT
  le_unitdata_req_t   *ludr;
  le_unitdata_req_t   *uudr;
  vlan_tci_t          *pci;
#elif isVLAN
  le_unitdata_req_t   *ludr;
  le_unitdata_req_t   *uudr;
  u16                 *pci;
#else
#error "Erro interno, verifique os fontes"
#endif

  unode = (unode_t *)q->q_ptr;
  if ( unode->uState != DL_IDLE ) {
    DEBUG_MSG_0( "##MOD##.ProcessaUnitDataReq: OUTSTATE" );
    DevolveUderrorInd( q, mp, DL_OUTSTATE, 0 );
    return;
  }
  lnode = unode->lnode;
  uudr = (void *)ludr = (void *)mp->b_rptr;

  if ( uudr->dl.dl_dest_addr_length != sizeof (uudr->dst) ) {
    DEBUG_MSG_0( "##MOD##.ProcessaUnitDataReq: BADADDR" );
    DevolveUderrorInd( q, mp, DL_BADADDR, 0 );
    return;
  }

  Size = msgdsize( mp ) + PCI_SIZE;

  #if DIAGNOSTIC
  if ( Size > lnode->iacCache->Ack.dl.info_ack.dl_max_sdu ) {
    DEBUG_MSG_0( "##MOD##.ProcessaUnitDataReq: BADDATA" );
    DevolveUderrorInd( q, mp, DL_BADDATA, 0 );
    return;
  }
  #endif

  /*  Monta o primeiro bloco da primitiva, com os argumentos.  Como      */
  /*  isto é apenas um pass-through DLPI-DLPI, posso simplesmente        */
  /*  copiar o bloco e modificar o que for necessário.  Com isso, o      */
  /*  processo de "priority-keeping" é automaticamente implementado.     */
  /*  Esta seção deve ser verificada com muito cuidado quando for feita */
  /*  uma nova implementação.                                            */

  /*  Obs: "priority-keeping": mantém o mesmo nível de prioridade que    */
  /*  foi usado na hora de passar a primitiva adiante.  Isso vale tanto */
  /*  para o HIPRI como para o priority band.                            */

  Sap = uudr->dst.sap;
```

```
  ludr->dl.dl_dest_addr_length = sizeof (ether_addr_t) + sizeof(ludr->dst.sap);
  /*  Esses comandos aqui não precisam ser executados.  Como o bloco    */
  /*  foi aproveitado, os dados já estão no lugar certo.  Isso também   */
  /*  funciona por causa do alinhamento de dados nas estruturas usadas. */
#if 0
  ludr->dst.ether = uudr->dst.ether;
  ludr->dl.dl_dest_addr_offset = uudr->dl.dl_dest_addr_offset;
#endif

//  Monta a próxima primitiva UNITDATA_REQ.
//    Aproveitando o bloco de memória anterior, para ser mais rápido.

#if   isLLC
  /*  Para o MAC 802.3, o campo SAP deve ser o tamanho do pacote.  (?)  */
  ludr->dst.sap = Size;

  /*  XXX - Arruma espaco para o novo M_PROTO.  Eu deveria realocar o   */
  /*  bloco, mas eu sei (?) que sempre vai ter espaco, então apenas     */
  /*  incremento o ponteiro.  Cuidado ao usar isso em um nova           */
  /*  implementação.                                                    */
  mp->b_wptr += 1;
#elif isSNAP
  ludr->dst.sap = SAP_SNAP;
#elif isVLT
  ludr->dst.sap = ETHERTYPE_VLAN;
#elif isVLAN
  /*  O driver VLT ignora este dado.                                    */
  /* ludr->dst.sap = ???;                                               */
#else
#error "Erro interno, verifique os fontes"
#endif

  /*  Construir  a LPCI.                                                */

  /*  Atualmente, aloca um novo bloco, mas pode ser feito               */
  /*  usando alocação maxima.  Não vou me preocupar com isso            */
  /*  agora.                                                            */
  /*  XXX -> Melhor opção: Aproveitar o próprio bloco que me            */
  /*  foi passado...  Preciso reservar espaco para isso.                */
  /*  Verificar o so_wroff da configuração do stream head               */
  if ( ( tmp = allocb( PCI_SIZE, BPRI_MED ) ) == NULL ) {
    /*  Sem memória, ignorar o pacote...                                */
    DEBUG_MSG_0( "##MOD##.ProcessaUnitDataReq: No memory" );
    freemsg( mp );
    return;
  }

  /* DB_TYPE(tmp) = M_DATA;  allocb() ja me devolve M_DATA      */
  tmp->b_wptr += PCI_SIZE;
  pci = (void *)tmp->b_rptr;
#if   isLLC
  pci->DSap = Sap;
  pci->SSap = unode->uSap;
  pci->Cmd  = LLC_UI;
```



```
#elif isSNAP
  pci->Org[ 0 ] = pci->Org[ 1 ] = pci->Org[ 2 ] = 0;
  pci->Code[ 0 ] = ( Sap >> 8 ) & 0xFF;
  pci->Code[ 1 ] = Sap & 0xFF;
#elif isVLT
  pci->bitmap.pri = PriStreamsToVlan( mp->b_band );
  pci->bitmap.cfi = 0;
  pci->bitmap.vid = unode->uSap;  //  Não tem como burlar a VLAN
#elif isVLAN
  *pci = Sap;
#else
#error "Erro interno, verifique os fontes"
#endif

  //  Montar e enviar a LPDU, a partir das partes construidas.
  tmp->b_cont = mp->b_cont;
  mp->b_cont = tmp;

  PUTNEXT( lnode->wrq, mp );
}

/**********************************************************************\
|*  A função ProcessaUnitDataInd é diferente demais para fazer o diff   *|
|*  por partes.  Deixar funções completamente independentes...          *|
\**********************************************************************/

#if isLLC

__PRIVATE__ __INLINE__ void
ProcessaUnitDataInd( lnode_t *lnode, mblk_t *mp )
{
  le_unitdata_ind_t *dui;
  llc1_unitdata_ind_t *lui;
  llc_pci_tipo1_t *pci;
  u8 Response, SSap, Poll, Cmd, DSap;
  unode_t *unode;

#if DIAGNOSTIC
  int Size;
#endif

  dui = (le_unitdata_ind_t *)mp->b_rptr;
  lui = (llc1_unitdata_ind_t *)mp->b_rptr;

#if DIAGNOSTIC
  /*  Antes de mais nada, trunca a mensagem no tamanho  */
  /* especificado pelo campo LENGTH (802.3) do quadro.  */

  Size = MBLKL(mp->b_cont);   //  XXX - E se b_cont for NULL ?
  if ( Size != dui->dst.sap )
  {
    //  Opa...  Não está correto, ajustar...
    DEBUG_MSG_2( "##MOD##.ProcessaUnitDataInd: Size mismatch: %d, %d",
        Size, dui->dst.sap
```

```
    );
    if ( Size > dui->dst.sap ) {
      if ( Size == 46 ) {
        /*  Pacote com padding.  Normal, ajusta.                        */
        mp->b_cont->b_wptr = mp->b_cont->b_rptr + dui->dst.sap;
      }
      else {
        DEBUG_MSG_2( "##MOD##.ProcessaUnitDataInd: Mensagem muito longa: "
                     "%d em vez de %d bytes",
                     Size, dui->dst.sap );
        freemsg( mp );
        return;
      }
    }
    else { /* Size < dui->dl.dst.sap */
      DEBUG_MSG_2( "##MOD##.ProcessaUnitDataInd: Mensagem muito curta: "
                   "%d em vez de %d bytes",
                   Size, dui->dst.sap );
      freemsg( mp );
      return;
    }
  }
#endif

  //  Verifica o LSAP destino.

  pci       = (llc_pci_tipo1_t *)mp->b_cont->b_rptr;
  Response  = pci->SSap &  LLC_RESPONSE;
  SSap      = pci->SSap & ~LLC_RESPONSE;
  Poll      = pci->Cmd  &  LLC_POLL;
  Cmd       = pci->Cmd  & ~LLC_POLL;
  DSap      = pci->DSap;

  if ( ( unode = FindClient( lnode, DSap ) ) != NULL ) {

#if 0
cmn_err( CE_NOTE, "Pacote com SAP 0x%02X para cliente 0x%08X", DSap, unode );
#endif

    /*  Mensagem para o usuário, passar adiante.                        */
    /*  No momento, só processo mensagens do tipo UI.                   */
    if ( Cmd == LLC_UI ) {

      //  Header: Colocar os Saps recebidos
      //  XXX - Estou assumindo formato conhecido no de_unitdata_ind_t
      lui->stuff1 = lui->stuff2 = 0;
      lui->src.sap = SSap;
      lui->dst.sap = DSap;

      //  Dados: Remover a PCI.
      mp->b_cont->b_rptr += sizeof (llc_pci_tipo1_t);

      //  Coloca na fila de saída do usuário
      PUTNEXT( unode->rdq, mp );
```



```
      }
      else {
        //  Comando inválido para o usuário...
        DEBUG_MSG_1( "##MOD##.ProcessaUnitDataInd: Comando LLC invalido (%x)",
                     Cmd
        );
        freemsg( mp );
      }
    }
  else if ( DSap == SAP_NULL ) {
    //  Mensagem para o próprio LLC station.
    DEBUG_MSG_0( "##MOD##: Recebi uma mensagem para o LSAP NULL" );
    DEBUG_MSG_1( "SAP Origem: %x", ((u8*)mp->b_cont->b_rptr)[1] );
    freemsg( mp );
  }
  else if ( DSap == SAP_GLOBAL ) {
    if ( pci->SSap == SAP_GLOBAL ) {
      //  Pacote Novell 802.3.  Ignorar !
      #if 0
      DEBUG_MSG_0( "Recebi um pacote Novell 802.3" );
      #endif
      freemsg( mp );
    }
    else {
      //  Mensagem global.
      DEBUG_MSG_0( "##MOD##.ProcessaUnitDataInd: "
                   "Recebi uma mensagem para o LSAP GLOBAL" );
      DEBUG_MSG_1( "SAP Origem: %x", ((u8*)mp->b_cont->b_rptr)[1] );
      freemsg( mp );
    }
  }
  else {
    //  Mensagem sem destinatário presente.
#if DEBUG > 2
    DEBUG_MSG_1( "##MOD##.ProcessaUnitDataInd: "
                 "Mensagem para LSAP desconhecido (0x%02X)", DSap );
#endif
    freemsg( mp );
  }
}

#elif isSNAP

__PRIVATE__ __INLINE__ void
ProcessaUnitDataInd( lnode_t *lnode, mblk_t *mp )
{
  le_unitdata_ind_t *sui;
  snap_pci_t *pci;
  unode_t *unode;
  u16 type;

  sui  = (le_unitdata_ind_t *)mp->b_rptr;
  pci  = (snap_pci_t*)mp->b_cont->b_rptr;
  type = ( pci->Code[0] << 8 ) + pci->Code[1];
```

```
  if ( ( unode = FindClient( lnode, type ) ) != NULL ) {

    //  Achei um cliente deste typespec

    //  Header: Colocar o typespec recebido
    //  XXX - Estou assumindo formato conhecido no llc1_unitdata_ind_t
    sui->src.sap = type;
    sui->dst.sap = type;

    //  Dados: Remover a PCI.
    mp->b_cont->b_rptr += sizeof (snap_pci_t);

    //  Coloca na fila de saida do usuário
    PUTNEXT( unode->rdq, mp );
  }
  else {
    //  Mensagem sem destinatário presente.
    DEBUG_MSG_1( "##MOD##.ProcessaUnitDataInd: "
                 "Mensagem para SNAP desconhecido (0x%04X)", type );
    freemsg( mp );
  }
}

#elif isVLT

__PRIVATE__ __INLINE__ void
ProcessaUnitDataInd( lnode_t *lnode, mblk_t *mp )
{
  le_unitdata_ind_t *sui;
  vlan_tci_t *tci;
  unode_t *unode;

  sui  = (le_unitdata_ind_t *)mp->b_rptr;
  tci  = (vlan_tci_t*)mp->b_cont->b_rptr;

  if ( tci->bitmap.cfi ) {
    cmn_err( CE_WARN, "##MOD##.ProcessaunitDataReq: "
                      "Ignorando pacote com bit CFI setado..." );
    freemsg( mp );
    return;
  }

  if ( ( unode = FindClient( lnode, tci->bitmap.vid  ) ) != NULL ) {

    //  Achei um cliente desta VLAN

    //  Header: Não preciso mudar NADA.  Uhu !!!

    //  Dados: Remover a PCI.
    mp->b_cont->b_rptr += PCI_SIZE;  //  Bypassa o tci

    //  Extra: Ajustar Band Priority na recepção
    mp->b_band = PriVlanToStreams( tci->bitmap.pri );
```



```
    //  Coloca na fila de saida do usuário
    PUTNEXT( unode->rdq, mp );
  }
  else {
    //  Mensagem sem destinatário presente.
    DEBUG_MSG_1( "##MOD##.ProcessaUnitDataInd: "
                 "Mensagem para VLT desconhecido (0x%04X)",
                 tci->bitmap.vid
    );
    freemsg( mp );
  }
}

#elif isVLAN

__PRIVATE__ __INLINE__ void
ProcessaUnitDataInd( lnode_t *lnode, mblk_t *mp )
{
  le_unitdata_ind_t *sui;
  u16 type;
  unode_t *unode;

  sui   = (le_unitdata_ind_t *)mp->b_rptr;
  type  = *(u16*)mp->b_cont->b_rptr;

  if ( ( unode = FindClient( lnode, type ) ) != NULL ) {

    // Achei um cliente deste ethertype.

    //  Header: Não preciso mudar NADA.  Uhu !!!

    //  Dados: Remover a PCI.
    mp->b_cont->b_rptr += PCI_SIZE;

    //  Coloca na fila de saida do usuário
    PUTNEXT( unode->rdq, mp );
  }
  else {
    //  Mensagem sem destinatário presente.
    DEBUG_MSG_1( "##MOD##.ProcessaUnitDataInd: "
                 "Mensagem para VLAN desconhecida (0x%04X)", type );
    freemsg( mp );
  }
}

#else
#error "Erro interno, verifique os fontes"
#endif

/*************************************************************************/

__PRIVATE__ __INLINE__ void
ProcessaInfoReq( queue_t *q, mblk_t *mp )
```

```
{
  mblk_t *tmp;
  unode_t *unode;
  lnode_t *lnode;
#if   isLLC
  llc1_info_ack_t *ack;
#elif isSNAP
  le_info_ack_t   *ack;
#elif isVLT
  le_info_ack_t   *ack;
#elif isVLAN
  le_info_ack_t   *ack;
#else
#error "Erro interno, verifique os fontes"
#endif

  unode = (unode_t *)q->q_ptr;

  tmp = ReallocMsgBlock( mp, sizeof (*ack), BPRI_MED );
  if ( tmp == NULL ) {
    cmn_err( CE_WARN, "##MOD##.ProcessaInfoReq: No memory..." );
    return;
  }
  DB_TYPE(tmp) = M_PCPROTO;
  tmp->b_wptr += sizeof (dl_info_ack_t) +       /*  Primitive   */
                 sizeof (ack->addr)     +       /*  Address     */
                 sizeof (ether_addr_t);         /*  BroadCast   */
                 /* != *_info_ack_t, por causa dos ulongs       */

  ack = (void *)tmp->b_rptr;
  ack->dl.dl_primitive  = DL_INFO_ACK;
  ack->dl.dl_min_sdu    = 0;
  ack->dl.dl_mac_type   = DL_CSMACD;
  ack->dl.dl_reserved   = 0;
  ack->dl.dl_current_state      = unode->uState;
  ack->dl.dl_sap_length = -sizeof( ack->addr.sap );
  ack->dl.dl_service_mode = DL_CLDLS;
  ack->dl.dl_qos_length = 0;
  ack->dl.dl_qos_offset = 0;
  ack->dl.dl_qos_range_length = 0;
  ack->dl.dl_qos_range_offset = 0;
  ack->dl.dl_provider_style= DL_STYLE2;
  ack->dl.dl_addr_offset = OFFSET( ack, &(ack->addr) );
  ack->dl.dl_version    = 2;
  ack->dl.dl_brdcst_addr_length = sizeof (ether_addr_t);
  ack->dl.dl_brdcst_addr_offset = OFFSET( ack, &(ack->bcast) );
  ack->dl.dl_growth = 0;
  ack->dl.dl_addr_length = sizeof (ack->addr);

  lnode = unode->lnode;
  if ( lnode == NULL || lnode->iacCache == NULL ) {
    cmn_err( CE_WARN, "##MOD##: DL_INFO_REQ with iacCache not initialized" );

    //  Assumindo defaults da ethernet.  Alguma ideia melhor ???
```



```
    ack->dl.dl_max_sdu  = MAX_MSG;
    ack->addr.sap = 0;
    ether_copy( &PrivateData.DefaultEtherAddr, &ack->addr.ether );
    ether_copy( &EtherBCast, &ack->bcast );
  }
  else {
    ack->dl.dl_max_sdu = lnode->iacCache->Ack.dl.info_ack.dl_max_sdu - PCI_SIZE;

    //  XXX - O que responder se estiver DL_UNBOUND ?
    ether_copy( lnode->iacCache->MyAddr, &ack->addr.ether );
    ack->addr.sap = unode->uSap;
    ether_copy( (lnode->iacCache->Ack.data +
                lnode->iacCache->Ack.dl.info_ack.dl_brdcst_addr_offset ),
                &ack->bcast );
  }
  qreply( q, tmp );
}

/*************************************************************************/

__PRIVATE__ __INLINE__ void
ProcessaLinkReq( queue_t *q, mblk_t *mp )
{
  struct linkblk *link;
  lnode_t *lnode;

  link = (struct linkblk *)mp->b_cont->b_rptr;
  /* enter mutex */

  #if DIAGNOSTIC
  //  Teoricamente, o kernel toma conta disso pra mim...
  if ( ( lnode = FindLNode( link->l_qbot ) ) != NULL ) {
    cmn_err( CE_WARN, "##MOD##:ProcessaLinkReq: Tentativa de relink" );
    DevolveIoctl( q, mp, M_IOCNAK, EFAULT, -1 );
    /* leave mutex */
    return;
  }
  #endif

  if ( ( lnode = GetNewLNode( link->l_qbot ) ) == NULL ) {
    // No more lower nodes...
    cmn_err( CE_WARN, "##MOD##: No more lnodes" );
    DevolveIoctl( q, mp, M_IOCNAK, ENOSR, -1 );
    /* leave mutex */
    return;
  }

  #if 0
  lnode->RefCount = 0;
  lnode->iacCache = NULL;
  bzero( (caddr_t)lnode->clients, sizeof (lnode->clients) );
  #else
  bzero( (caddr_t)lnode, sizeof( *lnode ) );
  #endif
```

```
  lnode->wrq        = link->l_qbot;
  lnode->rdq        = RD(link->l_qbot);
  lnode->rdq->q_ptr = lnode->wrq->q_ptr = lnode;

  /* leave mutex */

  DEBUG_MSG_0( "##MOD##:ProcessaLinkReq: Pedindo o InfoReq para o lnode" );
  lnode->mpCache = mp;
  lnode->qCache  = q;
  EnviaInfoReq( lnode->wrq );

  //  O resto do processamento é feito em ProcessaInfoAck()
}

/**********************************************************************/

__PRIVATE__ __INLINE__ void
ProcessaUnlinkReq( queue_t *q, mblk_t *mp )
{
  /*  XXX - Ainda tem muito o que fazer por aqui...                   */
  struct linkblk *link;
  unode_t *unode;
  lnode_t *lnode;
  client_t *cp;
  int i;

  unode = (unode_t *)q->q_ptr;

  /********************************************************************\
  |* IPC: Should not fail an unlink                                  *|
  \********************************************************************/

  link = (struct linkblk *)mp->b_cont->b_rptr;
  if ( ( lnode = FindLNode( link->l_qbot ) ) == NULL ) {
    //  Oops...
    cmn_err( CE_WARN, "I_UNLINK: lnode not found" );
    DevolveIoctl( q, mp, M_IOCNAK, EINVAL, -1 );
  }

  DEBUG_MSG_1( "##MOD##:ProcessaUnlinkReq: unlinking 0x%08X", lnode );

  if ( lnode->RefCount != 0 ) {
    //  OPA !!!  Dei unlink num lnode em uso !!!
    cp = &lnode->clients[0];
    for ( i = 0 ; i < MAX_CLIENTS ; ++i, ++cp ) {
      if ( cp->unode != NULL ) {
        DoHangup( cp->unode );
        DEBUG_MSG_1( "##MOD##: Hangup unode 0x%08X", cp->unode );
        #if DIAGNOSTIC
        cp->unode = NULL;
        #endif
      }
    }
  }
```



```
  /* XXX Ainda falta coisa aqui...  */
  lnode->wrq = lnode->rdq = NULL;
  kmem_free( lnode->iacCache, lnode->iacCache->Size );
  lnode->iacCache = NULL;

  DevolveIoctl( q, mp, M_IOCACK, 0, 0 );
}

/**********************************************************************/

__PRIVATE__ __INLINE__ void
ProcessaAttachReq( queue_t *q, mblk_t *mp, ulong ppa )
{
  unode_t *unode;
  lnode_t *lnode;

  unode = (unode_t *)q->q_ptr;
  if ( unode->uState != DL_UNATTACHED ) {
    cmn_err( CE_WARN, "##MOD##: ATTACH -> OUSTATE" );
    DevolveErrorAck( q, mp, DL_ATTACH_REQ, DL_OUTSTATE, 0 );
    return;
  }
  if ( ppa > MAX_LNODES ) {
    cmn_err( CE_WARN, "##MOD##:ProcessaAttachReq: PPA out of range" );
    DevolveErrorAck( q, mp, DL_ATTACH_REQ, DL_BADPPA, 0 );
    return;
  }
  lnode = &lnodelist[ ppa ];

  if ( lnode->iacCache == NULL ) {
    cmn_err( CE_WARN, "##MOD##:ProcessaAttachReq: PPA not initialized" );
    DevolveErrorAck( q, mp, DL_ATTACH_REQ, DL_BADPPA, 0 );
    return;
  }

  /* enter mutex */
  if ( AddClient( lnode, unode ) != 0 ) {
    DEBUG_MSG_1( "##MOD##.ProcessaAttachReq: lnode client unavailable, ppa = 0x%
02X", ppa );
    DevolveErrorAck( q, mp, DL_ATTACH_REQ, DL_SYSERR, ENOSR );
    return;
  }
  else {
    unode->lnode = lnode;
    lnode->RefCount++;
    unode->uState = DL_UNBOUND;
    DEBUG_MSG_1( "##MOD##: Attached to ppa #%d", lnode - lnodelist );
    DevolveOkAck( q, mp, DL_ATTACH_REQ );
  }
  /* leave mutex */
}

/**********************************************************************/
```

```
__PRIVATE__ __INLINE__ void
ProcessaDettachReq( queue_t *q, mblk_t *mp )
{
  unode_t *unode;

  unode = (unode_t *)q->q_ptr;
  if ( unode->uState != DL_UNBOUND ) {
    cmn_err( CE_WARN, "##MOD##: DETACH -> OUSTATE" );
    DevolveErrorAck( q, mp, DL_DETACH_REQ, DL_OUTSTATE, 0 );
    return;
  }

  DoDetach( unode );

  DevolveOkAck( q, mp, DL_DETACH_REQ );
}

/**********************************************************************\
|* Em DLPI o conceito de multicast é apenas a nível de endereço físico  *|
|* logo não podemos fazer nenhum tratamento especial para multicast de  *|
|* SAP.  Este pode ser pedido com um BIND no endereco desejado, ou com  *|
|* modo promíscuo, se for realmente necessário.                         *|
\**********************************************************************/

__PRIVATE__ __INLINE__ void
ProcessaEnabMultiReq( queue_t *wq, mblk_t *mp )
{
  unode_t *unode;

  CheckSize( wq, mp, DL_ENABMULTI_REQ_SIZE, DL_ENABMULTI_REQ );

  unode = (unode_t *)wq->q_ptr;
  if ( unode->uState == DL_UNATTACHED ) {
    cmn_err( CE_WARN, "##MOD##: ENABMULTIREQ -> OUSTATE" );
    DevolveErrorAck( wq, mp, DL_ENABMULTI_REQ, DL_OUTSTATE, 0 );
    return;
  }

  /*  XXX -- BACALHAU APENAS PARA TESTAR !!! */
  cmn_err( CE_WARN, "Usando feature multicast bacalhau !" );
  putnext( unode->lnode->wrq, mp );
  DevolveOkAck( wq, copymsg(mp), DL_ENABMULTI_REQ );
}

/**********************************************************************/

__PRIVATE__ __INLINE__ void
ProcessaDisabMultiReq( queue_t *q, mblk_t *mp )
{
  unode_t *unode;

  unode = (unode_t *)q->q_ptr;
  if ( unode->uState == DL_UNATTACHED ) {
    cmn_err( CE_WARN, "##MOD##: DISABMULTIREQ -> OUSTATE" );
```



```
    DevolveErrorAck( q, mp, DL_DISABMULTI_REQ, DL_OUTSTATE, 0 );
    return;
  }

  /*  XXX -- BACALHAU APENAS PARA TESTAR !!! */
  cmn_err( CE_WARN, "Usando feature bacalhau !" );
  putnext( unode->lnode->wrq, mp );
  DevolveOkAck( q, copymsg(mp), DL_DISABMULTI_REQ );
}

/**********************************************************************/

__PRIVATE__ __INLINE__ void
ProcessaPromiscOnReq( queue_t *wq, mblk_t *mp )
{
  CheckSize( wq, mp, DL_PROMISCON_REQ_SIZE, DL_PROMISCON_REQ );
  switch ( ((dl_promiscon_req_t*)mp->b_rptr)->dl_level ) {
    case DL_PROMISC_PHYS:
      DEBUG_MSG_0( "##MOD##: Recebi um PROMISCON_REQ(PHYS) (unsupported)" );
      break;
    case DL_PROMISC_SAP:
      DEBUG_MSG_0( "##MOD##: Recebi um PROMISCON_REQ(SAP) (unsupported)" );
      break;
    case DL_PROMISC_MULTI:
      DEBUG_MSG_0( "##MOD##: Recebi um PROMISCON_REQ(MULTI) (unsupported)" );
      break;
    default:
  }
  DevolveErrorAck( wq, mp, DL_PROMISCON_REQ, DL_UNSUPPORTED, 0 );
}

/**********************************************************************/

__PRIVATE__ __INLINE__ void
ProcessaPromiscOffReq( queue_t *wq, mblk_t *mp )
{
  CheckSize( wq, mp, DL_PROMISCOFF_REQ_SIZE, DL_PROMISCOFF_REQ );
  switch ( ((dl_promiscoff_req_t*)mp->b_rptr)->dl_level ) {
    case DL_PROMISC_PHYS:
      DEBUG_MSG_0( "##MOD##: Recebi um PROMISCOFF_REQ(PHYS) (unsupported)" );
      break;
    case DL_PROMISC_SAP:
      DEBUG_MSG_0( "##MOD##: Recebi um PROMISCOFF_REQ(SAP) (unsupported)" );
      break;
    case DL_PROMISC_MULTI:
      DEBUG_MSG_0( "##MOD##: Recebi um PROMISCOFF_REQ(MULTI) (unsupported)" );
      break;
    default:
  }
  DevolveErrorAck( wq, mp, DL_PROMISCOFF_REQ, DL_UNSUPPORTED, 0 );
}
```

```
/**************************************************************************\
|*  Module procedures                                                     *|
\**************************************************************************/

__PRIVATE__ int
##MOD##_open( queue_t *q, dev_t *devp, int flag, int sflag, cred_t *credp )
{
  unode_t *unode;
  minor_t device;

  if ( q->q_ptr ) {
    cmn_err( CE_WARN, "##MOD##.open: Isso não deveria acontecer !!!" );
    return EBUSY;
  }
  if ( sflag != CLONEOPEN ) {
    cmn_err( CE_WARN, "##MOD##: open without CLONE !" );
    return ENXIO;
  }

  /*  enter mutex  */
  for ( device = 0 ; device < MAX_UNODES ; device++ ) {
    if ( unodelist[device].rdq == NULL ) break;
  }
  *devp = makedevice( getmajor( *devp ), device );
  unode = &unodelist[ device ];
  WR(q)->q_ptr  = q->q_ptr = (char *)unode;
  unode->rdq    = q;
  unode->wrq    = WR(q);
  unode->uSap   = SAP_UNBOUND;
  unode->uState = DL_UNATTACHED;
  unode->lnode  = NULL;
  unode->uHang  = 0;
  /*  close mutex  */

  DEBUG_MSG_3( "Abrindo '##MOD##', q = 0x%08X, unode = 0x%08X, device = %d",
        q, unode, device );
  return 0;
}

/**************************************************************************/

__PRIVATE__ int
##MOD##_close( queue_t *q, int flag, cred_t *credp )
{
  unode_t *unode;

  unode = (unode_t *)q->q_ptr;

  DEBUG_MSG_2( "Fechando '##MOD##', q = 0x%08X, device = %d",
              q, unode - unodelist );
  DEBUG_MSG_2( "unode = 0x%08X, lnode = 0x%08X", unode, unode->lnode );

  switch ( unode->uState ) {
    case DL_IDLE:
```



```
      DoUnbind( unode );
      /*  fall through  */

    case DL_UNBOUND:
      DoDetach( unode );
      /*  fall through  */

    case DL_UNATTACHED:
      unode->rdq = unode->wrq = NULL;
      q->q_ptr   = NULL;
      break;

    default:
      cmn_err( CE_PANIC, "##MOD##.close: Invalid State" );
      cmn_err( CE_WARN, "Oops, o que eu estou fazendo aqui ???" );
      return 0;
  }

  return 0;
}

/**************************************************************************/

int
##MOD##_uwsvc( queue_t *q )
{
  /*  Upper Write Service -> Fila de Entrada da Escrita                  */
  mblk_t *mp;
  lnode_t *lnode;
  union DL_primitives *dlp;

  /* XXX - Hummmmmm.... */
  lnode = ((unode_t *)q->q_ptr)->lnode;
  while ( ( mp = getq( q ) ) != NULL ) {
    if ( lnode != NULL && !bcanputnext( lnode->wrq, mp->b_band ) ) {
      /*  O lnode existe e sua fila está cheia.  Volta...                */
      /*  Obs: Se o lnode não existir, deixa tratar mais adiante.        */
      DEBUG_MSG_0( "##MOD##_uwsvc: Next queue is full !" );
      putbq( q, mp );
      return 0;
    }
    dlp = (union DL_primitives*)mp->b_rptr;
    switch ( dlp->dl_primitive ) {
      default:
        cmn_err( CE_WARN, "##MOD##.Wsvc: Primitiva DLPI invalida (0x%X)",
          dlp->dl_primitive );
        freemsg( mp );
        break;

      case DL_INFO_REQ:
        DEBUG_MSG_0( "##MOD##: Recebi um INFO_REQ" );
        ProcessaInfoReq( q, mp );
        break;
```

```
      case DL_ATTACH_REQ:
        DEBUG_MSG_1( "##MOD##: Recebi um DL_ATTACH_REQ, PPA = %d",
                        dlp->attach_req.dl_ppa );
        ProcessaAttachReq( q, mp, dlp->attach_req.dl_ppa );
        break;

      case DL_BIND_REQ:
        DEBUG_MSG_1( "##MOD##: Recebi um BIND_REQ, SAP = 0x%04X",
                        dlp->bind_req.dl_sap );
        ProcessaBindReq( q, mp );
        break;

      case DL_UNBIND_REQ:
        DEBUG_MSG_0( "##MOD##: Recebi um UNBIND_REQ" );
        ProcessaUnbindReq( q, mp );
        break;

      case DL_DETACH_REQ:
        DEBUG_MSG_0( "##MOD##: Recebi um DETACH_REQ" );
        ProcessaDettachReq( q, mp );
        break;

      case DL_UNITDATA_REQ:
        #if DEBUG > 2
        DEBUG_MSG_0( "##MOD##: Recebi um UNITDATA_REQ" );
        #endif
        ProcessaUnitDataReq( q, mp );
        break;

      case DL_PHYS_ADDR_REQ:
        DEBUG_MSG_0( "##MOD##: Recebi um PHYS_ADDR_REQ" );
        ProcessaPhysAddrReq( q, mp );
        break;

      case DL_DISABMULTI_REQ:
        DEBUG_MSG_0( "##MOD##: Recebi um DISABMULTI_REQ (unsupported)" );
        ProcessaDisabMultiReq( q, mp );
        break;

      case DL_ENABMULTI_REQ:
        DEBUG_MSG_0( "##MOD##: Recebi um ENABMULTI_REQ (unsupported)" );
        ProcessaEnabMultiReq( q, mp );
        break;

      case DL_PROMISCON_REQ:
        ProcessaPromiscOnReq( q, mp );
        break;

      case DL_PROMISCOFF_REQ:
        ProcessaPromiscOffReq( q, mp );
        break;

      case DL_SET_PHYS_ADDR_REQ:
        DEBUG_MSG_0( "##MOD##: Recebi um SET_PHYS_ADDR_REQ (unsupported)" );
```



```
        DevolveErrorAck( q, mp, dlp->dl_primitive, DL_UNSUPPORTED, 0 );
        break;

      case DL_SUBS_BIND_REQ:
        DEBUG_MSG_0( "##MOD##: Recebi um SUBS_BIND_REQ (unsupported)" );
        DevolveErrorAck( q, mp, dlp->dl_primitive, DL_UNSUPPORTED, 0 );
        break;

      case DL_SUBS_UNBIND_REQ:
        DEBUG_MSG_0( "##MOD##: Recebi um SUBS_UNBIND_REQ (unsupported)" );
        DevolveErrorAck( q, mp, dlp->dl_primitive, DL_UNSUPPORTED, 0 );
        break;

      case DL_UDQOS_REQ:
        DEBUG_MSG_0( "##MOD##: Recebi um DL_UDQOS_REQ (unsupported)" );
        DevolveErrorAck( q, mp, dlp->dl_primitive, DL_UNSUPPORTED, 0 );
        break;
    }
  }

  return 0;
}

__PRIVATE__ int
##MOD##_uwput( queue_t *q, mblk_t *mp )
{
  struct iocblk *ioc;
  unode_t *unode;

  unode = (unode_t *)q->q_ptr;
  switch ( DB_TYPE(mp) ) {
    case M_IOCTL:

      if ( unode->uHang ) {
        DEBUG_MSG_1( "##MOD##: Tentativa de acesso a unode em hangup (0x%08X)",
                unode );
        DevolveIoctl( q, mp, M_IOCNAK, ENXIO, -1 );
        return 0;
      }

      ioc = (struct iocblk *)mp->b_rptr;

      switch ( ioc->ioc_cmd ) {
        case -1:
          //  Não sei o que é isso, mas vem junto com um flush...
          DevolveIoctl( q, mp, M_IOCNAK, EINVAL, -1 );
          break;

        case I_PLINK:
          ProcessaLinkReq( q, mp );
          break;

        case I_PUNLINK:
          ProcessaUnlinkReq( q, mp );
```

```
        break;

      case I_LINK:
      case I_UNLINK:
        cmn_err( CE_WARN,
              "##MOD##.uwput: Links nao persistentes nao sao autorizados" );
        DevolveIoctl( q, mp, M_IOCNAK, EINVAL, -1 );
        break;

      case DL_IOC_HDR_INFO:
        cmn_err( CE_WARN, "##MOD##.uwput: Recebi um ioctl(DL_IOC_HDR_INFO)" );
        goto Invalid;

      case ND_GET:
      case ND_SET:
        cmn_err( CE_WARN, "##MOD##.uwput: ndd commands not supported (yet)" );
        goto Invalid;

      default:
        /*  driver must fail ioctl  */
        cmn_err( CE_WARN, "##MOD##.uwput: Primitiva STREAMS IOCTL invalida" );
      Invalid:
        cmn_err( CE_WARN, "Len: %d, Cmd: 0x%08X, Id: 0x%08X, Count: %d",
                          MBLKL(mp),
                          ioc->ioc_cmd, ioc->ioc_id, ioc->ioc_count );
        DevolveIoctl( q, mp, M_IOCNAK, EINVAL, -1 );
        break;
    }
    break;

  case M_FLUSH:
    DEBUG_MSG_0( "Recebi um FLUSH em ##MOD##.uwput" );
    if ( *mp->b_rptr & FLUSHR ) {
      flushq( RD(q), FLUSHDATA );

      /*  É o lado de cima de um device driver, tem que devolver...   */
      *mp->b_rptr &= ~FLUSHW;
      qreply( q, mp );
    }
    else freemsg( mp );
    break;

  case M_PCPROTO:
  case M_PROTO:
    if ( unode->uHang ) {
      DEBUG_MSG_1( "##MOD##: Tentativa de acesso a unode em hangup (0x%08X)",
              unode );
      DevolveErrorMsg( q, mp, ENXIO, ENXIO );
      return 0;
    }
    putq( q, mp );
    break;

  case M_DATA:
```



```
    default:
      cmn_err( CE_WARN, "##MOD##.uwput: Primitiva STREAMS invalida (0x%x)",
        DB_TYPE(mp) );
      DevolveErrorMsg( q, mp, NOERROR, EINVAL );
      break;
  }
  return 0;
}

/*********************************************************************/

__PRIVATE__ int
##MOD##_lrput( queue_t *q, mblk_t *mp )
{
  /*  Lower Read Put -> Recebe os pacotes da camada inferior            */
  switch ( DB_TYPE(mp) ) {
    default:
      cmn_err( CE_WARN, "##MOD##.lrput: Primitiva STREAMS invalida (%x)",
        DB_TYPE(mp) );
      freemsg( mp );
      break;

    case M_FLUSH:
      DEBUG_MSG_0( "Recebi um FLUSH em ##MOD##.lrput" );
      /*  Não tenho uma queue de entrada, logo, não preciso fazer isso...
      if ( *mp->b_rptr & FLUSHW ) flushq( q, FLUSHALL );
      */
      /* Sou o lado de baixo de um driver, tenho que jogar isso fora... */
      freemsg( mp );
      break;

    case M_PCPROTO:
    case M_PROTO:
      //  Primitivas vindo da camada inferior
      switch ( ((union DL_primitives *)mp->b_rptr)->dl_primitive ) {
        default:
          cmn_err( CE_WARN, "##MOD##.lrput: Recebi uma primitiva invalida !" );
          freemsg( mp );
          break;

        case DL_INFO_ACK:
          DEBUG_MSG_0( "##MOD##.lrput: Recebi um INFO_ACK" );
          ProcessaInfoAck( (lnode_t *)(q->q_ptr), mp );
          break;

        case DL_UNITDATA_IND:
          ProcessaUnitDataInd( (lnode_t *)(q->q_ptr), mp );
          break;

        case DL_OK_ACK:
        case DL_ERROR_ACK:
          /*  Na verdade, eu deveria capturar e interpretar.            */
          /*  Pode ser que seja alguma resposta pra mim.                */
          cmn_err( CE_WARN, "Recebi um OK/ERROR_ACK" );
```

```
          freemsg( mp );
          break;
      }
      break;
  }
  return 0;
}

/************************************************************************/

__PRIVATE__ int
##MOD##_lwsvc( queue_t *q )
{
  /*  Lower Write Service -> Fila de Saida da Escrita                   */
  /*  Função unica: Acordar as filas de entrada...                      */
  int i;
  client_t *cp;

  cp = &((lnode_t *)q->q_ptr)->clients[0];
  for ( i = 0 ; i < MAX_CLIENTS ; ++i ) {
    if ( cp->unode != NULL ) {
      qenable( cp->unode->wrq );
#if DEBUG > 2
      cmn_err( CE_WARN, "lwsvc: Waking up unode 0x%08x", cp->unode );
#endif
    }
    ++cp;
  }
  return 0;
}

/************************************************************************/

__PRIVATE__ int
##MOD##_ursvc( queue_t *q )
{
  /*  Upper Read Service -> Fila de Saida da Leitura                    */
  /*  Função unica: Buffer de saida                                     */
  mblk_t *mp;

  while ( ( mp = getq( q ) ) != NULL ) {
    if ( bcanputnext( q, mp->b_band ) ) {
      putnext( q, mp );
    }
    else {
      putbq( q, mp );
      DEBUG_MSG_0( "##MOD##_ursvc: Next queue is full !" );
      return 0;
    }
  }
  return 0;
}
```



```
/************************************************************************\
|*  Module load functions                                               *|
\************************************************************************/

int
_init( void )
{
  int i;
  DEBUG_MSG_0( "##MOD##: Sendo carregado para o kernel..." );

  //  Aqui, garantir que todas as variaveis são inicializadas (?)
  bzero( (caddr_t)unodelist, sizeof (unodelist) );
  bzero( (caddr_t)lnodelist, sizeof (lnodelist) );

  i = mod_install( &modlinkage );
  return i;
}

int
_fini( void )
{
  DEBUG_MSG_0( "##MOD##: Removendo da memoria..." );
  return mod_remove( &modlinkage );
}

int
_info( struct modinfo *modinfop )
{
  DEBUG_MSG_0( "##MOD##: Alguem pediu uma informacao..." );
  return mod_info( &modlinkage, modinfop );
}

__PRIVATE__ int
##MOD##_getinfo( dev_info_t *dip, ddi_info_cmd_t cmd, void *arg, void **result )
{
  DEBUG_MSG_0( "##MOD##: Passei pelo getinfo !" );

  switch ( cmd ) {
    case DDI_INFO_DEVT2DEVINFO:
      *result = PrivateData.dip;
      return DDI_SUCCESS;
      break;

    case DDI_INFO_DEVT2INSTANCE:
      *result = (void *)getminor( (dev_t)arg );
      return DDI_SUCCESS;
      break;

    default:
      return DDI_FAILURE;
  }
}

__PRIVATE__ int
```

```
##MOD##_identify( dev_info_t *dip )
{
  DEBUG_MSG_0( "##MOD##: Passei pelo identify !" );

  if ( strcmp( ddi_get_name( dip ), modname ) == 0 ) {
    return DDI_IDENTIFIED;
  } else {
    return DDI_NOT_IDENTIFIED;
  }
}

__PRIVATE__ int
##MOD##_attach( dev_info_t *dip, ddi_attach_cmd_t cmd )
{
  extern int localetheraddr(struct ether_addr *, struct ether_addr *);

  DEBUG_MSG_0( "##MOD##: Passei pelo attach !" );

  if ( cmd != DDI_ATTACH ) return DDI_FAILURE;

  if ( ddi_create_minor_node( dip,
                              modname,
                              S_IFCHR,
                              0,
                              DDI_PSEUDO,
                              CLONE_DEV )
       == DDI_FAILURE )
  {
    ddi_remove_minor_node( dip, NULL );
    return DDI_FAILURE;
  }
  localetheraddr( NULL, &PrivateData.DefaultEtherAddr );
  PrivateData.dip = dip;
  ddi_set_driver_private( dip, (caddr_t)&PrivateData );
  return DDI_SUCCESS;
}

__PRIVATE__ int
##MOD##_detach( dev_info_t *dip, ddi_detach_cmd_t cmd )
{
  DEBUG_MSG_0( "##MOD##: Passei pelo detach !" );
  if ( cmd != DDI_DETACH ) return DDI_FAILURE;
  ddi_remove_minor_node( dip, NULL );
  PrivateData.dip = NULL;
  return DDI_SUCCESS;
}


/***********************************************************************\
|*  T't't't'that's all folks...                                        *|
\***********************************************************************/
```



```
/***********************************************************************\

Alguma documentação:

Posição de chamada das rotinas:


_info           -> A qualquer instante, mesmo antes do _init
_init           -> Na carga do módulo
_fini           -> Na descarga do módulo, antes do detach

identify        -> A qualquer instante, depois de carregado
attach          -> Antes de qualquer acesso
detach          -> No final de tudo, inclusive depois do _fini
getinfo         -> A qualquer instante, depois de attachado

open            -> Com o módulo attachado
close           -> Com o módulo attachado

...

Ha uma diferenca entre um usuário e um plumber ?
Como eu registro que um lnode pode ter mais de um usuário ?

Eu tenho que permitir o umplumb, logo deve ser permitido remover o
lnode com o unode aberto.  Tenho que tratar todos os erros de lnode
inexistente.  Não posso assumir em momento algum que o lnode existe
de forma persistente.

O identificador do plumber é um usuário unode valido ?  Acho que sim.
E por default, ele deve ser attachado ao lnode que ele criou.  Note-se
que isso só é possivel por causa das peculiaridades deste driver.  Não
é necessariamente valido em outros tipos de driver.

\***********************************************************************/
```

mux/vlan.h


```
#ifndef GTA_VLAN_H
#define GTA_VLAN_H

/**************************************************************************/

typedef union vlan_tci_t {
  struct {
    unsigned pri:3;
    unsigned cfi:1;
    unsigned vid:12;
  } bitmap;
  unsigned short s;
} vlan_tci_t;

/**************************************************************************/

#endif
```

mux/plumber.h


```
#define DEV_SNAP        "/dev/gtasnap"
#define DEV_LLC         "/dev/llc1mux"
#define DEV_VLT         "/dev/vlt"
#define DEV_VLAN        "/dev/vlan"
#define DEV_MAC         "/dev/le"
#define DEV_LOOP0       "/dev/gtalb0a"
#define DEV_LOOP1       "/dev/gtalb0b"

#define VLAN_DEFAULT    (1)
#define VLAN_JONNY      (1970)

#define PLUMB_LLCSNAP   1       //  Monta a estrutura de LLC/SNAP
#define PLUMB_VLAN      1       //  Monta a estrutura de VLAN
#define PLUMB_LOOPBACK  1       //  Usa o gtalb no plumbing
#define PLUMB_RETURN    0       //  Retorno do loopback, gasta mais CPU
```

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/file.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <stropts.h>
#include <net/if.h>
#include  <netinet/in_systm.h>
#include <netinet/in.h>
#include <netinet/if_ether.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <netdb.h>
#include <fcntl.h>
#include <sys/dlpi.h>
#include <signal.h>
#include <sys/systeminfo.h>
#include <string.h>

#include "dlpi.h"
#include "lib.h"
#include "gtallc1.h"
#include "llc.h"
#include "plumber.h"

int mac, llc, snap, vlt, vlan;
int muxid;


/*                                                                     *\

Estrutura de plumbing a ser criada:


le0     -> llc0 -> snap0
gtalb0a -> llc1 -> snap1
gtalb0b -> llc2 -> snap2  - Nao utilizado, para evitar gastos de CPu no lback.

\*                                                                     */

int
SafeOpen( const char *dev )
{
  int fd;

#if DEBUG
  printf( "Acessando dispositivo %s\n", dev );
#endif

  if ( ( fd = open( dev, O_RDWR ) ) < 0 ) {
```



```
    printf( "Erro: Nao pude acessar %s.\n", dev );
    perror( dev );
    exit( 1 );
  }

#if DEBUG
  printf( "Display Info %s\n", dev );
  DlpiDisplayFullInfo( fd );
#endif

  return fd;
}

void
AttachAndBind( int fd, int ppa, int sap
#if DEBUG
                                       , const char *dev
#endif
                                                          )
{
  DlpiAttach( fd, ppa );
  DlpiBind( fd, sap );

#if DEBUG
  printf( "Display Info %s, depois do bind\n", dev );
  DlpiDisplayFullInfo( fd );
#endif
}


int
Plumb( int above, int below )
{
  int muxid;

  if ( ( muxid = ioctl( above, I_PLINK, below ) ) < 0 ) {
    printf( "\nErro: Nao consegui contruir o multiplexer.\n" );
    perror( "ioctl(I_PLINK)" );
    exit( 1 );
  }

  return muxid;
}

void
UnplumbAll( int above )
{
  if ( ioctl( above, I_PUNLINK, MUXID_ALL ) < 0 ) {
    printf( "\nErro: Nao consegui destruir o multiplexer.\n" );
    perror( "ioctl(I_PUNLINK)" );
    exit( 1 );
  }
}
```

```
void
DoPlumb( void )
{

#if PLUMB_LLCSNAP
  llc = SafeOpen( DEV_LLC );

  mac = SafeOpen( DEV_MAC );
  AttachAndBind( mac, 0, ETHERTYPE_8023 );
  muxid = Plumb( llc, mac );
  printf( "llc1mux0->le0 (%d), ", muxid ); fflush( stdout );
  close( mac );

#if PLUMB_LOOPBACK

  mac = SafeOpen( DEV_LOOP0 );
  /*  AttachAndBind( mac, 0, ETHERTYPE_8023 );  */
  muxid = Plumb( llc, mac );
  printf( "llc1mux1->gtalb0a (%d), ", muxid ); fflush( stdout );
  close( mac );

#if PLUMB_RETURN

  mac = SafeOpen( DEV_LOOP1 );
  /*  AttachAndBind( mac, 0, ETHERTYPE_8023 );  */
  muxid = Plumb( llc, mac );
  printf( "llc1mux2->gtalb0b (%d), ", muxid ); fflush( stdout );
  close( mac );

#endif
#endif

  close( llc );

  /*******************************************************************/
  /*******************************************************************/

  snap = SafeOpen( DEV_SNAP );

  llc = SafeOpen( DEV_LLC );
  AttachAndBind( llc, 0, SAP_SNAP );
  muxid = Plumb( snap, llc );
  printf( "gtasnap0->llc1mux0 (%d), ", muxid ); fflush( stdout );
  close( llc );

#if PLUMB_LOOPBACK

  llc = SafeOpen( DEV_LLC );
  AttachAndBind( llc, 1, SAP_SNAP );
  muxid = Plumb( snap, llc );
  printf( "gtasnap1->llc1mux1 (%d), ", muxid ); fflush( stdout );
  close( llc );

#if PLUMB_RETURN
```



```
  llc = SafeOpen( DEV_LLC );
  AttachAndBind( llc, 2, SAP_SNAP );
  muxid = Plumb( snap, llc );
  printf( "gtasnap2->llc1mux2 (%d), ", muxid ); fflush( stdout );
  close( llc );

#endif
#endif

  close( snap );

#endif PLUMB_LLCSNAP

#if PLUMB_VLAN

  vlt = SafeOpen( DEV_VLT );
  mac = SafeOpen( DEV_MAC );
  AttachAndBind( mac, 0, ETHERTYPE_VLAN );
  muxid = Plumb( vlt, mac );
  printf( "vlt0->le0 (%d), ", muxid ); fflush( stdout );
  close( mac );
  close( vlt );

  vlan = SafeOpen( DEV_VLAN );
  vlt  = SafeOpen( DEV_VLT );
  AttachAndBind( vlt, 0, VLAN_DEFAULT );
  muxid = Plumb( vlan, vlt );
  printf( "vlan0(vid=%d)->vlt0 (%d), ", VLAN_DEFAULT, muxid ); fflush( stdout );
  close( vlt );
  close( vlan );

  vlan = SafeOpen( DEV_VLAN );
  vlt  = SafeOpen( DEV_VLT );
  AttachAndBind( vlt, 0, VLAN_JONNY );
  muxid = Plumb( vlan, vlt );
  printf( "vlan1(vid=%d)->vlt0 (%d), ", VLAN_JONNY, muxid ); fflush( stdout );
  close( vlt );
  close( vlan );

#endif PLUMB_VLAN
}

void
DoUnplumb( void )
{

#if PLUMB_LLCSNAP
  snap = SafeOpen( DEV_SNAP );
  UnplumbAll( snap );
  printf( "gtasnap, " );  fflush( stdout );
  close( snap );

  llc = SafeOpen( DEV_LLC );
```

```
  UnplumbAll( llc );
  printf( "llc1mux, " );  fflush( stdout );
  close( llc );
#endif

#if PLUMB_VLAN
  vlan = SafeOpen( DEV_VLAN );
  UnplumbAll( vlan );
  printf( "vlan, " );  fflush( stdout );
  close( vlan );

  vlt = SafeOpen( DEV_VLT );
  UnplumbAll( vlt );
  printf( "vlt, " );  fflush( stdout );
  close( vlt );
#endif

}

void
Usage( void )
{
  printf( "Usage: plumber on|off\n" );
}

int
main( int argc, const char **argv )
{
  int WillPlumb;

  if ( argc != 2 ) {
    Usage();
    return 1;
  }

  if ( !strcasecmp( argv[1], "on" ) ) {
    WillPlumb = 1;
  }
  else if ( !strcasecmp( argv[1], "off" ) ) {
    WillPlumb = 0;
  }
  else {
    Usage();
    return 1;
  }

  printf( "Syncing." ); /*  Paranoid mode                               */
  fflush( stdout );
  sync();
  printf( "." );
  fflush( stdout );
  sync();
  printf( "." );
  fflush( stdout );
```



```
  sync();
  printf( "\n\n" );

  DlpiVerbose = 0;
  if ( WillPlumb ) {
    printf( "Plumbing: " );
    fflush( stdout );
    DoPlumb();
  }
  else {
    printf( "Unplumbing: " );
    fflush( stdout );
    DoUnplumb();
  }
  printf( "ok...\n" );

  return 0;
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/file.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <stropts.h>
#include <net/if.h>
#include  <netinet/in_systm.h>
#include <netinet/in.h>
#include <netinet/if_ether.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <netdb.h>
#include <fcntl.h>
#include <sys/dlpi.h>
#include <signal.h>
#include <sys/systeminfo.h>
#include <string.h>
#include <inet/led.h>
#include <sys/signal.h>
#include <sys/fault.h>
#include <sys/syscall.h>
#include <sys/procfs.h>
#include <sys/wait.h>

#include "dlpi.h"
#include "lib.h"
#include "gtallc1.h"
#include "llc.h"
#include "rtprio.h"

#include "plumber.h"

int dd;
struct strbuf MsgCtrl, MsgData;
dl_u MsgCtrlBuffer;
char MsgDataBuffer[2048];


int
main( int argc, char * const *argv )
{
  DlpiVerbose = 0;

#if PLUMB_LLCSNAP
  if ( ( dd = open( DEV_SNAP, O_RDWR ) ) < 0 ) {
    perror( DEV_SNAP );
    exit( 1 );
  }
  DlpiAttach( dd, 0 );
```



```
  DlpiDetach( dd );
  close( dd );

  if ( ( dd = open( DEV_LLC, O_RDWR ) ) < 0 ) {
    perror( DEV_LLC );
    exit( 1 );
  }
  DlpiAttach( dd, 0 );
  DlpiDetach( dd );
  close( dd );
#endif

#if PLUMB_VLAN
  if ( ( dd = open( DEV_VLT, O_RDWR ) ) < 0 ) {
    perror( DEV_VLT );
    exit( 1 );
  }
  DlpiAttach( dd, 0 );
  DlpiDetach( dd );
  close( dd );

  if ( ( dd = open( DEV_VLAN, O_RDWR ) ) < 0 ) {
    perror( DEV_VLAN );
    exit( 1 );
  }
  DlpiAttach( dd, 0 );
  DlpiDetach( dd );
  close( dd );
#endif

  return 0;
}
```

mux/start


```
#! /bin/sh

if [ "$1" = "" ] ; then
  set `hostname`
fi

case "$1" in
  parati)
    ip=parati.snap
    ;;
  buzios)
    ip=buzios.snap
    ;;
  theta)
    ip=theta.snap
    ;;
  *)
    echo "Opcao nao suportada"
    exit
    ;;
esac

set -x

./checkplumb > /dev/null || plumber on
ifconfig gtasnap0 plumb
ifconfig gtasnap0 inet $ip netmask 255.255.255.0
ifconfig gtasnap0 up
arp -s buzios.snap 08:00:20:75:67:86
arp -s parati.snap 08:00:20:76:8f:ff
arp -s theta.snap  00:00:3b:80:21:45

ifconfig gtasnap1 plumb
ifconfig gtasnap1 inet 10.127.0.1 netmask 255.255.255.0
ifconfig gtasnap1 up
arp -s 10.127.0.2 00:10:4b:22:7e:54

if [ "$1" != "buzios" ] ; then
  #  Para permitir o acesso continuado ao NFS da ROMA, com a rota antiga
  route add host 146.164.53.65 angra 1
  route add default fundao.snap 1
  route delete default angra
fi
```

mux/stop


```
#! /bin/sh

set -x

route add default angra 1
route delete default fundao.snap

ifconfig gtasnap0 unplumb
ifconfig gtasnap1 unplumb

plumber off
```

```
#include <sys/types.h>
#include <sys/stream.h>
#include <sys/stropts.h>
#include <sys/param.h>
#include <sys/ddi.h>
#include <sys/modctl.h>

extern void printf( const char *, ... );
extern char * sprintf( const char *, ... );

#define MULTITHREAD_SAFE        0

#define         DEBUG           1

#if DEBUG
#define DEBUG_MSG_0(Msg)        printf(Msg)
#define DEBUG_MSG_1(Msg,a)      printf(Msg,a)
#else
#define DEBUG_MSG_0(Msg)
#define DEBUG_MSG_1(Msg,a)
#endif

extern struct fmodsw gtadebug_fsw;

static struct module_info gtadebug_info = {
  0x0,
  gtadebug_fsw.f_name,
  0,            /*  min msgsize                                         */
  INFPSZ,       /*  max msgsize                                         */
  512,          /*  hiwater                                             */
  128           /*  lowater                                             */
};

/*  Modules structures  */

static int      gtadebug_open(), gtadebug_close(),
                gtadebug_rput(), gtadebug_wput();

static struct qinit rinit = {
  gtadebug_rput,
  NULL,
  gtadebug_open,
  gtadebug_close,
  NULL,
  &gtadebug_info,
  NULL
};

static struct qinit winit = {
  gtadebug_wput,
  NULL,
  NULL,
  NULL,
  NULL,
```



```
  &gtadebug_info,
  NULL
};

static struct streamtab gtadebug_tab = { &rinit, &winit, NULL, NULL };

static struct fmodsw gtadebug_fsw = {
  "gtadebug",
  &gtadebug_tab,
#if MULTITHREAD_SAFE
  D_NEW|D_MP|D_MTOUTPERIM|D_MTOCEXCL
#else
  D_NEW
#endif
};

/*  Isso aqui e' extern mesmo.  Parece que e' colocado pelo modload !!! */
extern struct mod_ops mod_strmodops;

static struct modlstrmod modlstrmod = {
  &mod_strmodops,
  "GTA: Debug STREAMS Module",
  &gtadebug_fsw
};

static struct modlinkage modlinkage = {
  MODREV_1,
  {
    (void*)&modlstrmod,
    NULL
  }
};

/**********************************************************************/

static int InstanceNo = 0;

/**********************************************************************/
/*  Module private functions                                          */
/**********************************************************************/

/*  None defined                                                      */

/**********************************************************************/
/*  Module procedures                                                 */
/**********************************************************************/

#define MyInstance(q)   ((int)((q)->q_ptr))

static int
gtadebug_open( queue_t *q, dev_t *devp, int flag, int sflag, cred_t *credp )
{
  MyInstance(q) = MyInstance(WR(q)) = ++InstanceNo;
```

```
  printf( "Abrindo modulo GTADEBUG(%d)\n", MyInstance(q) );

#if MULTITHREAD_SAFE
  qprocson( q );
#endif
  return 0;
}

static const char *
MsgType( mblk_t *mp )
{
  static char Aux[10];
  int Type;
  static struct {
    int Val;
    const char *Name;
  } Lista[] = {
        { M_PCPROTO,    "M_PCPROTO"     },
        { M_PROTO,      "M_PROTO"       },
        { M_DATA,       "M_DATA"        },
        { M_IOCTL,      "M_IOCTL"       },
        { M_IOCACK,     "M_IOCACK"      },
        { M_IOCNAK,     "M_IOCNAK"      },
        { M_FLUSH,      "M_FLUSH"       },
        { -1,           NULL            }
  }, *lp;;

  Type = mp->b_datap->db_type;
  for ( lp = Lista ; lp->Name != NULL ; ++lp ) {
    if ( Type == lp->Val ) {
      return lp->Name;
    }
  }
  return sprintf( Aux, "0x%02X", mp->b_datap->db_type );
}

#include "debug/dlpi.dbg"

static int
gtadebug_wput( queue_t *q, mblk_t *mp )
{
  Processa( q, mp, "wput" );
  putnext( q, mp );
  return 0;
}

static int
gtadebug_rput( queue_t *q, mblk_t *mp )
{
  Processa( q, mp, "rput" );
  putnext( q, mp );
  return 0;
}
```



```
static int
gtadebug_close( queue_t *q, int flag, cred_t *credp )
{
  DEBUG_MSG_0( "Fechando modulo GTADEBUG\n" );
#if MULTITHREAD_SAFE
  qprocsoff( q );
#endif

  MyInstance(q) = -1;
  return 0;
}

/************************************************************************/
/*  Module load functions                                               */
/************************************************************************/

int
_init( void )
{
  DEBUG_MSG_0( "gtadebug: Sendo carregado para o kernel...\n" );
  return mod_install( &modlinkage );
}

int
_fini( void )
{
  DEBUG_MSG_0( "gtadebug: Removendo da memoria...\n" );
  return mod_remove( &modlinkage );
}

int
_info( struct modinfo *modinfop )
{
  DEBUG_MSG_0( "gtadebug: Alguem pediu uma informacao...\n" );
  return mod_info( &modlinkage, modinfop );
}

/************************************************************************/
/*  T't't't'that's all folks...                                         */
/************************************************************************/
```

```
#include <sys/dlpi.h>
#include <sys/ethernet.h>

#include "dlpi.h"

static const char *
DlpiPrimType( ulong Prim )
{
  switch ( Prim ) {
    case DL_INFO_REQ:
      return "DL_INFO_REQ";
    case DL_INFO_ACK:
      return "DL_INFO_ACK";
    case DL_ATTACH_REQ:
      return "DL_ATTACH_REQ";
    case DL_DETACH_REQ:
      return "DL_DETACH_REQ";
    case DL_BIND_REQ:
      return "DL_BIND_REQ";
    case DL_BIND_ACK:
      return "DL_BIND_ACK";
    case DL_UNBIND_REQ:
      return "DL_UNBIND_REQ";
    case DL_OK_ACK:
      return "DL_OK_ACK";
    case DL_ERROR_ACK:
      return "DL_ERROR_ACK";
    case DL_PHYS_ADDR_REQ:
      return "DL_PHYS_ADDR_REQ";
    case DL_PHYS_ADDR_ACK:
      return "DL_PHYS_ADDR_ACK";
    case DL_SET_PHYS_ADDR_REQ:
      return "DL_SET_PHYS_ADDR_REQ";
    case DL_UNITDATA_REQ:
      return "DL_UNITDATA_REQ";
    default:
      return "UNKNOWN";
  }
}

static void
Dump( u_char *p, int len )
{
  int n, i;

  n = 0;
  while ( 1 ) {
    i = 0;
    printf( "Dump %02X:", n++ );
    while ( 1 ) {
      printf( " %02X", *p++ );
      ++i;
      --len;
      if ( len == 0 ) {
```



```
        printf( "\n" );
        return;
      }
      if ( i == 16 ) {
        printf( "\n" );
        break;
      }
    }
  }
}

static void
Processa( queue_t *q, mblk_t *mp, const char *s )
{
  ulong Size;
  char Aux[80], *p;
  dl_u *dlp;

  Size = mp->b_wptr - mp->b_rptr;
  sprintf( Aux, "GTADEBUG(%d).%s: ", MyInstance(q), s);
  for ( p = Aux ; *p != 0 ; p++ );

  switch ( mp->b_datap->db_type ) {
    case M_PCPROTO:
    case M_PROTO:
      /*  Estou debugando a DLPI, vamos ver as primitivas !!!  */
      dlp = (void *)mp->b_rptr;
      switch ( dlp->dl.dl_primitive ) {
        default:
          sprintf( p, "DLPI %s", DlpiPrimType( dlp->dl.dl_primitive ) );
          for ( p = Aux ; *p != 0 ; p++ );
          printf( "%s\n", Aux );
          Dump( mp->b_rptr, Size );
          return;
      }
      break;

    case M_IOCTL:
    case M_IOCACK:
    case M_IOCNAK:
      sprintf( p, "IOCTL family" );
      for ( p = Aux ; *p != 0 ; p++ );
      printf( "%s\n", Aux );
      Dump( mp->b_rptr, Size );
      break;

    default:
      sprintf( p, "%s", MsgType( mp ) );
      for ( p = Aux ; *p != 0 ; p++ );
      printf( "%s\n", Aux );
      break;
  }
}
```

mux/debug/generic.dbg


```c
static void
Processa( queue_t *q, mblk_t *mp, const char *s )
{
  ulong Size;
  char Aux[80], *p;

  sprintf( Aux, "GTADEBUG(%d).%s: Pacote tipo %s, ",
        MyInstance(q),
        s,
        MsgType( mp )
  );
  for ( p = Aux ; *p != 0 ; p++ );
  Size = mp->b_wptr - mp->b_rptr;
  if ( Size >= 12 ) {
    sprintf( p, "Data(%lu): 0x%04lX, 0x%04lX, 0x%04lX", Size,
        ((ulong*)mp->b_rptr)[0],
        ((ulong*)mp->b_rptr)[1],
        ((ulong*)mp->b_rptr)[2]
    );
  }
  else if ( Size >= 8 ) {
    sprintf( p, "Data(%lu): 0x%04lX, 0x%04lX", Size,
        ((ulong*)mp->b_rptr)[0],
        ((ulong*)mp->b_rptr)[1]
    );
  }
  else if ( Size >= 4 ) {
    sprintf( p, "Data(%lu): 0x%04lX", Size,
        ((ulong*)mp->b_rptr)[0]
    );
  }
  printf( "%s\n", Aux );
}
```

mux/push


```sh
#! /bin/ksh

name2major()
{
  echo `awk '$1 == "'${1}'" { print $2; }' /etc/name_to_major`
}

namelist='gtasnap'

tmp=/tmp/push.$$
case "$1" in
  on)
    rm -f ${tmp}
    echo -n "setting autopush for:"
    for a in `echo $namelist` ; do
      echo -n " $a"
      echo "$a -1 0 gtadebug" >> ${tmp}
    done
    autopush -f ${tmp}
    echo ""
    rm -f ${tmp}
    ;;
  off)
    echo -n "removing autopush for:"
    for a in `echo $namelist` ; do
      echo -n " $a"
      autopush -r -M `name2major $a` -m 0
    done
    echo ""
    ;;
  *)
    echo "Usage: push on|off"
    ;;
esac
```

loopback/Makefile


```
LIBDIR  = ../lib
CC      = gcc
OPT     = -O6
CKFLAGS = -Wall ${OPT} -D_KERNEL -I${LIBDIR}

KERNEL  = /kernel
MODUTIL = ../util/modutil

default:        all

all:            gtalb gtalb.conf

asm:            gtalb.s

clean:
                -rm gtalb.conf
                -rm *.o gtalb

install:        ${KERNEL}/drv/gtalb

${KERNEL}/drv/gtalb:            gtalb
                cp -p gtalb $@
                ${MODUTIL} unload gtalb

reload:
                ${MODUTIL} unload gtalb

load:
                ../util/modprobe /dev/gtalb

gtalb:  gtalb.o
                ld -r -o gtalb gtalb.o
                chmod +x gtalb

gtalb.o:        gtalb.c
                ${CC} ${CKFLAGS} -c gtalb.c

gtalb.s:        gtalb.c
                ${CC} -S ${CKFLAGS} -c gtalb.c

create:         install
        cp -p gtalb.conf ${KERNEL}/drv/gtalb.conf
        add_drv -m '* 0660 root ether' gtalb

remove:
        -rem_drv gtalb
        -rm ${KERNEL}/drv/gtalb.conf

reinstall: remove create

gtalb.conf:     Makefile
        echo 'name="gtalb" parent="pseudo" instance=0;' > gtalb.conf
```

loopback/gtalb.c


```
/**********************************************************************\

GTALB - Loopback simulando ETHERNET_802.3

Obs: Stateless !

\**********************************************************************/

#include <sys/types.h>
#include <sys/stream.h>
#include <sys/stropts.h>
#include <sys/param.h>
#include <sys/conf.h>
#include <sys/ddi.h>
#include <sys/modctl.h>
#include <sys/errno.h>
#include <sys/sunddi.h>
#include <sys/kmem.h>
#include <sys/stat.h>
#include <sys/strsun.h>

#include <sys/socket.h>
#include <net/if.h>
#include <netinet/in.h>
#include <netinet/if_ether.h>
#include <sys/dlpi.h>
#include <sys/cmn_err.h>

#include "dlpi.h"
```

```
/**********************************************************************/
/*  Local definitions                                                 */
/**********************************************************************/

#define MULTITHREAD_SAFE        0
#define IGNORE_BROKEN_PIPES     1
#define IGNORE_BIND             0

#define DEBUG                   0
#define PULLUPMSG               1
#define MAXSDU                  1500


#if 0
#define __INLINE__      inline
#define __PRIVATE__     static
#else
#define __INLINE__
#define __PRIVATE__
#endif

#define MAXLINKS                (2)
#define MAXUNODES               (MAXLINKS*2)

#if DEBUG
#define DEBUG_MSG_0(Msg)        cmn_err(CE_NOTE,Msg)
#define DEBUG_MSG_1(Msg,a)      cmn_err(CE_NOTE,Msg,a)
#define DEBUG_MSG_2(Msg,a,b)    cmn_err(CE_NOTE,Msg,a,b)
#else
#define DEBUG_MSG_0(Msg)
#define DEBUG_MSG_1(Msg,a)
#define DEBUG_MSG_2(Msg,a,b)
#endif
#define DIAGNOSTIC DEBUG
```



```
/**********************************************************************/
/*  Module Configuration                                              */
/**********************************************************************/

__PRIVATE__ char modname[] = "gtalb";

__PRIVATE__ struct module_info gtalb_info = {
  0x1232,
  modname,
  0,            /*  min msgsize                                       */
  MAXSDU,       /*  max msgsize                                       */
  10240,        /*  hiwater                                           */
  1024          /*  lowater                                           */
};

/*  Modules structures  */

__PRIVATE__ int gtalb_open(),  gtalb_close(),
                gtalb_wput(),
                gtalb_rsvc(),
                gtalb_wsvc();


__PRIVATE__ int gtalb_getinfo( dev_info_t *, ddi_info_cmd_t, void *, void ** );
__PRIVATE__ int gtalb_identify( dev_info_t * );
__PRIVATE__ int gtalb_attach( dev_info_t *, ddi_attach_cmd_t );
__PRIVATE__ int gtalb_detach( dev_info_t *, ddi_detach_cmd_t );

__PRIVATE__ struct qinit rinit = {
  NULL,                 gtalb_rsvc,
  gtalb_open,           gtalb_close,
  NULL,                 &gtalb_info,
  NULL
};

__PRIVATE__ struct qinit winit = {
  gtalb_wput,           gtalb_wsvc,
  NULL,                 NULL,
  NULL,                 &gtalb_info,
  NULL
};

__PRIVATE__ struct streamtab gtalb_tab = { &rinit, &winit, NULL, NULL };

__PRIVATE__ struct cb_ops gtalb_cb_ops = {
  NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
  &gtalb_tab,
#if MULTITHREAD_SAFE
  D_MP
#else
  0
#endif
};
```

```
__PRIVATE__ struct dev_ops gtalb_ops = {
  DEVO_REV,
  0,
  gtalb_getinfo,
  gtalb_identify,
  nulldev,              /*  probe  */
  gtalb_attach,
  gtalb_detach,
  nodev,                /*  reset */
  &gtalb_cb_ops,
  NULL
};


__PRIVATE__ struct modldrv modldrv = {
  &mod_driverops,
  "GTA: Loopback Ethernet Driver",
  &gtalb_ops
};

__PRIVATE__ struct modlinkage modlinkage = { MODREV_1, { (void*)&modldrv, NULL }
 };


/************************************************************************/
/*  Module private data                                                 */
/************************************************************************/

typedef struct unode unode_t;

/*  Estrutura para definir cada um dos nodes                            */

struct unode {
  queue_t       *rdq;          /*  Backref to the unode                 */
  queue_t       *wrq;          /*  Backref to the unode                 */
  unode_t       *peer;         /*  Peer Node associated                 */
  ushort        uSap;          /*  Usado no pseudo bind                 */
};

__PRIVATE__ unode_t unodelist[ MAXUNODES ];
```



```
/************************************************************************/
/*  Module private funtions                                             */
/************************************************************************/

static inline void
        memcpy( void *To, void *From, size_t N ) { bcopy( From, To, N ); }

#if DIAGNOSTIC
#define freemsg(a)      \
                if (a) freemsg(a); \
                else cmn_err( CE_PANIC, "freemsg invalido: %d", __LINE__ )
#endif

__PRIVATE__ mblk_t *
ReallocMsgBlock( mblk_t *mp, int Size, uint Pri )
{
  /*  Tenta reaproveitar os blocos pre-alocados.                        */
  /*  IPC: Usa sintaxe de msg, mas trata como blk !!!                   */
  /*  Deve ser equivalente a um freemsg seguido de um allocb            */
  static ulong Hits = 0, Misses = 0;

  int OldSize = mp->b_datap->db_lim - mp->b_datap->db_base;

#if DEBUG
  cmn_err( CE_CONT,
    "Block Size: %d, Block Usage: %d, New Usage: %d, Hits: %d/%d\n",
    OldSize, mp->b_wptr - mp->b_rptr,
    Size, Hits, Hits + Misses
  );
#endif

  if ( OldSize >= Size && mp->b_datap->db_ref == 1 ) {
    /* Reaproveita.                                                     */
    ++Hits;

    mp->b_wptr = mp->b_rptr = mp->b_datap->db_base;
    #ifdef FULL_ALLOCB_COMPATIBILITY
    mp->b_datap->db_data = M_DATA;
    #endif
    if ( mp->b_cont != NULL ) {
      freemsg( mp->b_cont );
      mp->b_cont = NULL;
    }
    return mp;
  }
  else {
    freemsg( mp );
    ++Misses;
    return allocb( Size, Pri );
  }
}
```

```
/**********************************************************************/
/*  Funcoes de tratamento de mensagens                                */
/**********************************************************************/

__PRIVATE__ void
DevolveErrorAck( queue_t *q, mblk_t *mp, ulong Prim, ulong Error, ulong UnixErro
r )
{
  dl_error_ack_t *eap;

  mp = ReallocMsgBlock( mp, sizeof (dl_error_ack_t), BPRI_MED );
  if ( mp == NULL ) {
    cmn_err( CE_WARN, "gtalb.DevolveErrorAck: No memory..." );
    return;
  }
  mp->b_datap->db_type = M_PCPROTO;
  mp->b_wptr = mp->b_rptr + sizeof (dl_error_ack_t);
  eap = (dl_error_ack_t*)mp->b_rptr;
  eap->dl_primitive = DL_ERROR_ACK;
  eap->dl_error_primitive = Prim;
  eap->dl_errno = Error;
  eap->dl_unix_errno = UnixError;
  qreply( q, mp );
}

/**********************************************************************/

__PRIVATE__ void
DevolveUderrorInd( queue_t *q, mblk_t *mp, ulong Error, ulong UnixError )
{
  le_uderror_ind_t *ude;
  mblk_t *tmp;

  tmp = allocb( sizeof (le_uderror_ind_t), BPRI_MED );
  if ( tmp == NULL ) {
    cmn_err( CE_WARN, "gtalb.DevolveUderrorInd: No memory..." );
    freemsg( mp );
    return;
  }
  tmp->b_datap->db_type = M_PCPROTO;
  tmp->b_wptr = tmp->b_rptr + sizeof (le_uderror_ind_t);
  ude = (le_uderror_ind_t *)tmp->b_rptr;
  ude->dl.dl_primitive = DL_UDERROR_IND;
  ude->dl.dl_dest_addr_length = sizeof(le_ether_addr_t);
  ude->dl.dl_dest_addr_offset = OFFSET( ude, &(ude->addr) );
  ude->dl.dl_unix_errno = UnixError;
  ude->dl.dl_errno = Error;
  memcpy( &ude->addr, &((le_unitdata_req_t*)mp->b_rptr)->dst,
                                      sizeof (le_ether_addr_t) );
  qreply( q, tmp );
  freemsg( mp );
}

/**********************************************************************/
```



```
__PRIVATE__ void
DevolveOkAck( queue_t *q, mblk_t *mp, ulong Prim )
{
  dl_ok_ack_t *oap;

  mp = ReallocMsgBlock( mp, sizeof (dl_ok_ack_t), BPRI_MED );
  if ( mp == NULL ) {
    cmn_err( CE_WARN, "gtalb.DevolveOkAck: No memory..." );
    return;
  }
  mp->b_datap->db_type = M_PCPROTO;
  mp->b_wptr = mp->b_rptr + sizeof (dl_ok_ack_t);
  oap = (dl_ok_ack_t*)mp->b_rptr;
  oap->dl_primitive = DL_OK_ACK;
  oap->dl_correct_primitive = Prim;
  qreply( q, mp );
}

/**********************************************************************/

__PRIVATE__ __INLINE__ void
DevolveIoctl( queue_t * q, mblk_t *mp, u_char type, int error, int rval )
{
  struct iocblk *ioc;

  ioc = (struct iocblk *)mp->b_rptr;
  mp->b_datap->db_type = type;          /*  M_IOCACK || M_IOCNAK        */
  ioc->ioc_error = error;               /*  Error Code  (errno ?)       */
  ioc->ioc_rval = rval;                 /*  Return Value                */
  qreply( q, mp );
}

/**********************************************************************/

__PRIVATE__ void
DevolveErrorMsg( queue_t *q, mblk_t *mp, u_char RdError, u_char WrError )
{
  mblk_t *tmp;

  tmp = ReallocMsgBlock( mp, 2, BPRI_HI );

  if ( tmp == NULL ) {
    cmn_err( CE_WARN, "gtalb.DevolveErrorMsg: No memory..." );
    return;
  }
  mp->b_datap->db_type = M_ERROR;
  *mp->b_wptr++ = RdError;
  *mp->b_wptr++ = WrError;
  qreply( q, mp );
}
```

```
/*********************************************************************/
/*  Module Processing Funtions                                       */
/*********************************************************************/

__PRIVATE__ __INLINE__ void
ProcessaBindReq( queue_t *q, mblk_t *mp )
{
#if IGNORE_BIND
  /*  XXX: Mesmo ignorando, tenho que devolver um Ok...              */
#else
  DevolveErrorAck( q, mp, DL_BIND_REQ, DL_OUTSTATE, 0 );
#endif
}

/*********************************************************************/

__PRIVATE__ __INLINE__ void
ProcessaUnbindReq( queue_t *q, mblk_t *mp )
{
#if IGNORE_BIND
  /*  XXX: Mesmo ignorando, tenho que devolver um Ok...              */
#else
  DevolveErrorAck( q, mp, DL_BIND_REQ, DL_BUSY, 0 );
#endif
}

/*********************************************************************/


__PRIVATE__ __INLINE__ void
ProcessaUnitDataReq( queue_t *q, mblk_t *mp )
{
  unode_t *peer;
  le_unitdata_req_t *udr;
  le_unitdata_ind_t *udi;
  mblk_t *tmp;

  peer = ((unode_t *)q->q_ptr)->peer;
  udr = (le_unitdata_req_t*)mp->b_rptr;

  if ( udr->dl.dl_dest_addr_length != sizeof (le_ether_addr_t) ) {
    DEBUG_MSG_0( "gtalb.ProcessaUnitDataReq: BADADDR" );
    DevolveUderrorInd( q, mp, DL_BADADDR, 0 );
    return;
  }

#if PULLUPMSG
  /*  Realinha os dados, para simular uma interface de rede.         */
  /*  Deve ser feito mesmo que nao tenha peer, para ser real.  (???) */
  if ( !pullupmsg( mp->b_cont, -1 ) ) {
    freemsg( mp );
    return;
  }
#endif
```



```
  if ( peer ) {
    /*  Converte a mensagem le_unitdata_req em le_unitdata_ind           */
    tmp = allocb( sizeof (le_unitdata_ind_t), BPRI_MED );

    if ( tmp == NULL ) {
      cmn_err( CE_WARN, "gtalb.ProcessaUnitDataReq: No memory..." );
      freemsg( mp );
      return;
    }
    /*  Poderia tentar copiar o db_type da mensagem original, mas isso  */
    /*  não é possível no mundo real (ethernet), então não vou fazer    */
    /*  neste driver loopback.                                          */
    tmp->b_datap->db_type = M_PROTO;
    tmp->b_wptr = tmp->b_rptr + sizeof (le_unitdata_ind_t);

    udi = (le_unitdata_ind_t*)tmp->b_rptr;
    udi->dl.dl_primitive = DL_UNITDATA_IND;
    bzero( (caddr_t)&udi->src, sizeof (udi->src) );
    memcpy( &udi->dst, &udr->dst, sizeof (udi->dst) );

    tmp->b_cont = mp->b_cont;
    freeb( mp );
    if ( putq( peer->rdq, tmp ) != 1 ) {
      cmn_err( CE_NOTE, "gtalb.ProcessaUnitDataReq: erro na putq !!!" );
    }
  }
  else {
    #if IGNORE_BROKEN_PIPES
    freemsg( mp );
    #else
    DevolveUderrorInd( q, mp, DL_ACCESS, EPIPE );
    #endif
  }
}

/*********************************************************************/


__PRIVATE__ __INLINE__ void
ProcessaInfoReq( queue_t *q, mblk_t *mp )
{
  le_info_ack_t *ack;
  mblk_t *tmp;
  unode_t *unode;

  unode = (unode_t *)q->q_ptr;

  tmp = ReallocMsgBlock( mp, sizeof (le_info_ack_t), BPRI_MED );

  if ( tmp == NULL ) {
    cmn_err( CE_WARN, "gtalb.ProcessaInfoReq: No memory..." );
    return;
  }
  tmp->b_datap->db_type = M_PCPROTO;
  tmp->b_wptr = tmp->b_rptr + sizeof (dl_info_ack_t) +
```

```
                              sizeof (le_ether_addr_t) +
                              sizeof (ether_addr_t);
                              /* != le_info_ack_t, por causa dos ulongs */

  ack = (le_info_ack_t *)tmp->b_rptr;
  ack->dl.dl_primitive  = DL_INFO_ACK;

  ack->dl.dl_min_sdu    = 0;
  ack->dl.dl_mac_type   = DL_CSMACD;
  ack->dl.dl_reserved   = 0;
  ack->dl.dl_current_state = DL_IDLE;
  ack->dl.dl_sap_length = -2;
  ack->dl.dl_service_mode = DL_CLDLS;
  ack->dl.dl_qos_length = 0;
  ack->dl.dl_qos_offset = 0;
  ack->dl.dl_qos_range_length = 0;
  ack->dl.dl_qos_range_offset = 0;
  ack->dl.dl_provider_style= DL_STYLE1;
  ack->dl.dl_addr_offset = OFFSET( ack, &(ack->addr) );
  ack->dl.dl_version    = 2;
  ack->dl.dl_brdcst_addr_length = sizeof (ether_addr_t);
  ack->dl.dl_brdcst_addr_offset = OFFSET( ack, &(ack->bcast) );
  ack->dl.dl_growth = 0;
  ack->dl.dl_addr_length = sizeof (le_ether_addr_t);

  ack->dl.dl_max_sdu    = MAXSDU;

  bzero( (caddr_t)&ack->addr.ether, sizeof (ether_addr_t) );
  ack->addr.sap = 0;
  bzero( (caddr_t)&ack->bcast, sizeof (ether_addr_t) );

  qreply( q, tmp );
}
```



```
/**********************************************************************/
/*  Module procedures                                                 */
/**********************************************************************/

__PRIVATE__ int
gtalb_open( queue_t *q, dev_t *devp, int flag, int sflag, cred_t *credp )
{
  unode_t *unode, *peer;
  minor_t device;

  if ( q->q_ptr ) {
    /*  Already open !  */
    return EBUSY;
  }

  if ( sflag == CLONEOPEN ) {
    cmn_err( CE_WARN, "gtalb: open with CLONE !" );
    return ENXIO;
  }

  device = getminor( *devp );
  if ( device >= MAXUNODES ){
    return ENXIO;
  }

  if ( unodelist[ device ].rdq != NULL ) {
    cmn_err( CE_WARN, "gtalb: Device em uso nao preindentificado" );
    return EBUSY;
  }

  DEBUG_MSG_1( "gtalb_open: Minor alocated: %d", device );
  *devp = makedevice( getmajor( *devp ), device );

  unode = &unodelist[ device ];
  WR(q)->q_ptr  = q->q_ptr = (char *)unode;
  unode->rdq    = q;
  unode->wrq    = WR(q);

  /*  Verificar se o peer esta' aberto !                              */
  peer = &unodelist[ device ^ 0x01 ];
  if ( peer->rdq != NULL ) {
    unode->peer = peer;
    DEBUG_MSG_2( "gtalb: Estabelecendo link: 0x%08X <-> 0x%08X",
                 unode, unode->peer );
  }
  else {
    DEBUG_MSG_1( "gtalb: Abrindo sem link: 0x%08X", unode );
    unode->peer = NULL;
  }

  return 0;
}

/**********************************************************************/
```

```
__PRIVATE__ int
gtalb_close( queue_t *q, int flag, cred_t *credp )
{
  unode_t *unode;

  unode = (unode_t *)q->q_ptr;

  DEBUG_MSG_2( "Fechando modulo GTALB, q = 0x%08X, unode = 0x%08X", q, unode );

  if ( unode->peer ) {
    unode->peer->peer = NULL;
    unode->peer = NULL;
  }
  unode->rdq = unode->wrq = NULL;
  q->q_ptr   = NULL;

  return 0;
}

/*************************************************************************/

int
gtalb_wsvc( queue_t *q )
{
  /*  Upper Write Service -> Fila de Entrada da Escrita                 */
  mblk_t *mp;
  unode_t *peer;
  dl_u *dlp;

  peer = ((unode_t *)q->q_ptr)->peer;
  while ( peer == NULL || canput( peer->rdq ) ) {
    if ( ( mp = getq( q ) ) == NULL ) {
      return 0;
    }
    dlp = (dl_u*)mp->b_rptr;
    switch ( dlp->dl.dl_primitive ) {
      /*  Estou ignorando a prioridade aqui, pois nao uso filas */
      default:
        cmn_err( CE_WARN, "gtalb.Wsvc: Primitiva DLPI invalida (0x%X)",
          ((dl_u*)mp->b_rptr)->dl.dl_primitive );
        freemsg( mp );
        break;

      case DL_INFO_REQ:
        DEBUG_MSG_0( "gtalb: Recebi um INFO_REQ" );
        ProcessaInfoReq( q, mp );
        break;

      case DL_BIND_REQ:
        DEBUG_MSG_0( "gtalb: Recebi um BIND_REQ" );
        ProcessaBindReq( q, mp );
        break;
```



```
      case DL_UNBIND_REQ:
        DEBUG_MSG_0( "gtalb: Recebi um UNBIND_REQ" );
        ProcessaUnbindReq( q, mp );
        break;

      case DL_UNITDATA_REQ:
        ProcessaUnitDataReq( q, mp );
        break;

      case DL_ATTACH_REQ:
      case DL_DETACH_REQ:
      case DL_PROMISCON_REQ:
      case DL_PROMISCOFF_REQ:
      case DL_SUBS_BIND_REQ:
      case DL_SUBS_UNBIND_REQ:
      case DL_UDQOS_REQ:
      case DL_DISABMULTI_REQ:
      case DL_ENABMULTI_REQ:
      case DL_PHYS_ADDR_REQ:
        DevolveErrorAck( q, mp, dlp->dl.dl_primitive, DL_UNSUPPORTED, 0 );
        break;
    }
  }
  return 0;
}

__PRIVATE__ int
gtalb_wput( queue_t *q, mblk_t *mp )
{
  struct iocblk *ioc;
  unode_t *unode;

  unode = (unode_t *)q->q_ptr;
  switch ( mp->b_datap->db_type ) {
    case M_IOCTL:

      ioc = (struct iocblk *)mp->b_rptr;

      switch ( ioc->ioc_cmd ) {
        case -1:
          /*  Nao sei o que fazer, nem o que e' isso, mas vem junto     */
          /*  com um flush...                                           */
          DevolveIoctl( q, mp, M_IOCNAK, EINVAL, -1 );
          break;

        default:
          /*  driver must fail ioctl  */
          cmn_err( CE_WARN, "gtalb.wput: Primitiva STREAMS IOCTL invalida" );
          cmn_err( CE_WARN, "Len: %d, Cmd: 0x%08X, Id: 0x%08X, Count: %d",
                            mp->b_wptr - mp->b_rptr,
                            ioc->ioc_cmd,
                            ioc->ioc_id,
                            ioc->ioc_count );
```

```
          DevolveIoctl( q, mp, M_IOCNAK, EINVAL, -1 );
          break;
      }
      break;

    case M_FLUSH:
      DEBUG_MSG_0( "Recebi um FLUSH em gtalb.wput" );
      if ( *mp->b_rptr & FLUSHR ) {
        flushq( RD(q), FLUSHDATA );

        /*  E' o lado de cima de um device driver, tem que devolver...  */
        *mp->b_rptr &= ~FLUSHW;
        qreply( q, mp );
      }
      else freemsg( mp );
      break;

    case M_PCPROTO:
    case M_PROTO:
      putq( q, mp );
      break;

    case M_DATA:
    default:
      cmn_err( CE_WARN, "gtalb.wput: Primitiva STREAMS invalida (0x%x)",
        mp->b_datap->db_type );

      DevolveErrorMsg( q, mp, NOERROR, EINVAL );
      break;
  }
  return 0;
}

/************************************************************************/

__PRIVATE__ int
gtalb_rsvc( queue_t *q )
{
  mblk_t *mp;

  while ( canputnext( q ) ) {
    if ( ( mp = getq( q ) ) == NULL ) {
      return 0;
    }
    putnext( q, mp );
  }
  /*
  cmn_err( CE_NOTE, "gtalb.rsvc: Next queue is full !" );
  */
  return 0;
}
```



```
/**********************************************************************/
/*  Module load functions                                             */
/**********************************************************************/

int
_init( void )
{
  int i;
  DEBUG_MSG_0( "gtalb: Sendo carregado para o kernel..." );
  i = mod_install( &modlinkage );
  return i;
}

int
_fini( void )
{
  DEBUG_MSG_0( "gtalb: Removendo da memoria..." );
  return mod_remove( &modlinkage );
}

int
_info( struct modinfo *modinfop )
{
  DEBUG_MSG_0( "gtalb: Alguem pediu uma informacao..." );
  return mod_info( &modlinkage, modinfop );
}

__PRIVATE__ dev_info_t *gtalb_dev_info = NULL;

__PRIVATE__ int
gtalb_getinfo( dev_info_t *dip, ddi_info_cmd_t cmd, void *arg, void **result )
{
  DEBUG_MSG_0( "gtalb: Passei pelo getinfo !" );

  switch ( cmd ) {
    case DDI_INFO_DEVT2DEVINFO:
      *result = gtalb_dev_info;
      return DDI_SUCCESS;
      break;

    case DDI_INFO_DEVT2INSTANCE:
      *result = (void *)getminor( (dev_t)arg );
      return DDI_SUCCESS;
      break;

    default:
      return DDI_FAILURE;
  }
}

__PRIVATE__ int
gtalb_identify( dev_info_t *dip )
{
  DEBUG_MSG_0( "gtalb: Passei pelo identify !" );
```

```
  if ( strcmp( ddi_get_name( dip ), modname ) == 0 ) {
    return DDI_IDENTIFIED;
  } else {
    return DDI_NOT_IDENTIFIED;
  }
}

__PRIVATE__ int
gtalb_attach( dev_info_t *dip, ddi_attach_cmd_t cmd )
{
  int i;
  char devname[80];

  DEBUG_MSG_0( "gtalb: Passei pelo attach !" );

  if ( cmd != DDI_ATTACH ) return DDI_FAILURE;

  for ( i = 0 ; i < MAXLINKS ; ++i ) {
    sprintf( devname, "%s%da", modname, i );
    if ( ddi_create_minor_node( dip, devname, S_IFCHR, i*2, DDI_NT_NET, 0 )
         == DDI_FAILURE )
    {
      ddi_remove_minor_node( dip, NULL );
      return DDI_FAILURE;
    }
    sprintf( devname, "%s%db", modname, i );
    if ( ddi_create_minor_node( dip, devname, S_IFCHR, i*2+1, DDI_NT_NET, 0 )
         == DDI_FAILURE )
    {
      ddi_remove_minor_node( dip, NULL );
      return DDI_FAILURE;
    }
  }
  gtalb_dev_info = dip;
  return DDI_SUCCESS;
}

__PRIVATE__ int
gtalb_detach( dev_info_t *dip, ddi_detach_cmd_t cmd )
{
  DEBUG_MSG_0( "gtalb: Passei pelo detach !" );
  if ( cmd != DDI_DETACH ) return DDI_FAILURE;
  ddi_remove_minor_node( dip, NULL );
  gtalb_dev_info = NULL;
  return DDI_SUCCESS;
}


/*************************************************************************/
/*  T't't't'that's all folks...                                          */
/*************************************************************************/
```



```
#! /bin/make -f

CC      = gcc
CFLAGS  = -O -g -Wall -Wno-implicit
LFLAGS  = -lsocket -lnsl

MODULES = dlpi.o keyboard.o lib.o pr_ip.o libstream.o rtprio.o

all:    libgta.a

libgta.a:       ${MODULES}
        ar ru libgta.a ${MODULES}

clean:
        -rm -f *.o *.a core

dlpi.o:         dlpi.c dlpi.h
rtprio.o:       rtprio.c rtprio.h
keyboard.o:     keyboard.c keyboard.h
```

```
#ifndef GTA_DLPI_H
#define GTA_DLPI_H

extern int DlpiVerbose;                 /*  Default = 1 -> on           */

/*  Faltaram esses aqui no Solaris                                      */
#define ETHERTYPE_IPX   (0x8137)
#define ETHERTYPE_8023  (0x05DC)        /* 1500                         */
#define ETHERTYPE_VLAN  (0x8100)

#define SIZE_DL_U       ( sizeof (union DL_primitives) + 128 )

typedef union {
  union DL_primitives dl;
  char  data[SIZE_DL_U];        /*  Para inserir dados pos-escritos     */
} dl_u;


const char *StrState( int State );
const char *DlpiServiceModeName( unsigned long Mode );

#ifndef _KERNEL

void ShowInfoAck( dl_info_ack_t *p );
void ShowErrorAck( dl_error_ack_t *p );

void DlpiDisplayInfo( int dd );
void DlpiDisplayFullInfo( int dd );
void DlpiDisplayPhysAddr( int dd );
void DlpiAttach( int dd, int Port );
void DlpiBind( int dd, long sap );
void DlpiUnbind( int dd );
void DlpiDetach( int dd );
void DlpiPromiscuous( int dd, int Mode );
int  DlpiSendPacket( int dd, void *buffer, int buflen );
int  DlpiRecvPacket( int dd, void *buffer, int buflen );
ulong DlpiMaxSdu( int dd );

#endif

/**********************************************************************/

/*  Estas estruturas funcionam no /dev/le do Solaris 2.4                */

typedef struct le_ether_addr_t {
  struct ether_addr     ether;
  u_short               sap;            /*  ethertype ou 802.3 length   */
} le_ether_addr_t;

typedef struct le_unitdata_ind_t {
  dl_unitdata_ind_t dl;
  le_ether_addr_t dst;
  le_ether_addr_t src;
} le_unitdata_ind_t;
```



```
typedef struct le_unitdata_req_t {
  dl_unitdata_req_t dl;
  le_ether_addr_t dst;
} le_unitdata_req_t;

typedef struct le_uderror_ind_t {
  dl_uderror_ind_t dl;
  le_ether_addr_t  addr;
} le_uderror_ind_t;

typedef struct le_bind_ack_t {
  dl_bind_ack_t   dl;
  le_ether_addr_t addr;
} le_bind_ack_t;

typedef struct le_phys_addr_req_t {
  dl_phys_addr_req_t   dl;
  le_ether_addr_t      addr;
} le_phys_addr_req_t;

typedef struct le_phys_addr_ack_t {
  dl_phys_addr_ack_t   dl;
  le_ether_addr_t      addr;
} le_phys_addr_ack_t;

typedef struct le_info_ack_t {
  dl_info_ack_t     dl;
  le_ether_addr_t   addr;
  struct ether_addr bcast;
} le_info_ack_t;

/**********************************************************************/

#endif
```

```
/*

  DLPI.C - Algumas bibliotecas de apoio `a programacao da DLPI

*/

#include <errno.h>
#include <string.h>

#include <sys/stropts.h>
#include <sys/dlpi.h>
#include <sys/types.h>

#include "lib.h"
#include "dlpi.h"

struct ether_addr DlpiMyAddr;

int DlpiVerbose = 1;

const char *
StrState( int State )
{
  static char Unk[ 80 ];

  switch ( State ) {
    default:
      sprintf( Unk, "Unknown (0x%02X)", State );
      return Unk;
    case DL_UNBOUND:
      return "DL_UNBOUND";
    case DL_IDLE:
      return "DL_IDLE";
    case DL_UNATTACHED:
      return "DL_UNATTACHED";
  }
}


void
ShowInfoAck( dl_info_ack_t *p )
{
  /*
  printf( "Max SDU = %ld\n", p->dl_max_sdu );
  printf( "Mac Type = %ld\n", p->dl_mac_type );
  */
  printf( "Current State = %s\n", StrState( p->dl_current_state ) );
}

const char *
DlpiStyleName( unsigned long Style )
{
  if ( Style == DL_STYLE1 ) {
    return "DL_STYLE1";
```



```
  }
  else if ( Style == DL_STYLE2 ) {
    return "DL_STYLE2";
  }
  else {
    return "UNKNOWN";
  }
}

const char *
DlpiMacTypeName( unsigned long MacType )
{
  switch ( MacType ) {
    case DL_CSMACD:
      return "DL_CSMACD";
    case DL_ETHER:
      return "DL_ETHER";
    case DL_TPR:
      return "DL_TPR";
    case DL_FDDI:
      return "DL_FDDI";
    default:
      return "UNKNOWN";
  }
}

static int
GetDlpiInfo( int dd, dl_u *u )
{
  struct strbuf ctrl;
  dl_info_ack_t *a;
  int Flags;
  int i;

  u->dl.info_req.dl_primitive = DL_INFO_REQ;
  ctrl.buf = (void*)&u->dl.info_req;
  ctrl.maxlen = ctrl.len = sizeof u->dl.info_req;
  putmsg( dd, &ctrl, NULL, RS_HIPRI );

  Flags = RS_HIPRI;
  ctrl.buf = (void*)u;
  ctrl.len = ctrl.maxlen = sizeof *u;
Again:
  if ( ( i = getmsg( dd, &ctrl, NULL, &Flags ) ) != 0 ) {
    if ( errno == EINTR ) goto Again;  /* Por que ??? */
    perror( "getmsg" );
    printf( "i = %d\n", i );
    printf( "Prim = 0x%02lX\n", u->dl.dl_primitive );
    InternalError();
    exit( 1 );
  }

  a = &u->dl.info_ack;
```

```
  if ( a->dl_primitive != DL_INFO_ACK ) {
    printf( "Erro: Primitiva recebida invalida (%02lX)\n", a->dl_primitive );
    return -1;
  }

  if ( ( sizeof(dl_info_ack_t) + a->dl_addr_length +
         a->dl_brdcst_addr_length) != ctrl.len )
  {
    printf( "Warning: Extra data (%ld bytes)\n",
      ctrl.len -
      ( sizeof(dl_info_ack_t) + a->dl_addr_length + a->dl_brdcst_addr_length )
    );
    printf( "info_ack = %d, addr = %ld:%ld, bcast = %ld:%ld\n",
            sizeof (dl_info_ack_t),
            a->dl_addr_offset,
            a->dl_addr_length,
            a->dl_brdcst_addr_offset,
            a->dl_brdcst_addr_length );
    printf( "ctrl.len: %d\n", ctrl.len );
  }
  return 0;
}

void
DlpiDisplayInfo( int dd )
{
  dl_u u;

  if ( GetDlpiInfo( dd, &u ) == 0 ) {
    ShowInfoAck( &u.dl.info_ack );
  }
}

void
DlpiDisplayFullInfo( int dd )
{
  dl_u u;
  dl_info_ack_t *a;
  int i;

  if ( GetDlpiInfo( dd, &u ) != 0 ) {
    return;
  }
  a = &u.dl.info_ack;
  printf(      "Max SDU:                 %lu\n", a->dl_max_sdu );
  printf(      "Min SDU:                 %lu\n", a->dl_min_sdu );
  printf(      "Address Len:             %lu\n", a->dl_addr_length );
  printf(      "Mac Type:                %s\n", DlpiMacTypeName( a->dl_mac_type )
 );
  printf(      "Current State:           %s\n",  StrState( a->dl_current_state )
);
  printf(      "SAP Len:                 %ld\n",  a->dl_sap_length );
  printf(      "Service Mode:            %s\n",  DlpiServiceModeName( a->dl_servi
ce_mode ) );
```



```
  printf(      "QOS Values Len:         %lu\n", a->dl_qos_length );
  printf(      "QOS Range Len:          %lu\n", a->dl_qos_range_length );
  printf(      "QOS Range Off:          %lu\n", a->dl_qos_range_offset );
  printf(      "DLPI Style:             %s\n",  DlpiStyleName( a->dl_provider_st
yle ) );
  printf(      "DLSAP Addr Offset:      %lu\n", a->dl_addr_offset );
  printf(      "Version:                %lu\n", a->dl_version );
  printf(      "Broadcast Addr Len:     %lu\n", a->dl_brdcst_addr_length );
  printf(      "Broadcast Addr Off:     %lu\n", a->dl_brdcst_addr_offset );
  if ( a->dl_addr_offset != 0 ) {
    printf(      "Interface Addr:                  " );
    if ( a->dl_mac_type == DL_ETHER || a->dl_mac_type == DL_CSMACD ) {
      for ( i = 0 ; i < 6 ; ++i ) {
        if ( i != 0 ) printf( ":" );
        printf( "%02X", ((unsigned char*)a)[ a->dl_addr_offset + i ] );
      }
      if ( a->dl_addr_length == 8 ) {
        printf( ".0x%04X",
          ((unsigned char*)a)[ a->dl_addr_offset + 6 ] * 256 +
          ((unsigned char*)a)[ a->dl_addr_offset + 7 ]
        );
      }
      else {
        for ( i = 6 ; i < a->dl_addr_length ; ++i ) {
          if ( i != 0 ) printf( "." );
          printf( "%u", ((unsigned char*)a)[ a->dl_addr_offset + i ] );
        }
      }
    }
    else {
      for ( i = 0 ; i < a->dl_addr_length ; ++i ) {
        if ( i != 0 ) printf( "." );
        printf( "%u", ((unsigned char*)a)[ a->dl_addr_offset + i ] );
      }
    }
    printf( "\n" );
  }

  if ( a->dl_brdcst_addr_offset != 0 ) {
    printf(      "Broadcast Addr:                  " );
    if ( a->dl_mac_type == DL_ETHER || a->dl_mac_type == DL_CSMACD ) {
      for ( i = 0 ; i < 6 ; ++i ) {
        if ( i != 0 ) printf( ":" );
        printf( "%02X", ((unsigned char*)a)[ a->dl_brdcst_addr_offset + i ] );
      }
      if ( a->dl_brdcst_addr_length == 8 ) {
        printf( ".0x%04X",
          ((unsigned char*)a)[ a->dl_brdcst_addr_offset + 6 ] * 256 +
          ((unsigned char*)a)[ a->dl_brdcst_addr_offset + 7 ]
        );
      }
      else {
        for ( i = 6 ; i < a->dl_brdcst_addr_length ; ++i ) {
          if ( i != 0 ) printf( "." );
```

```
          printf( "%u", ((unsigned char*)a)[ a->dl_brdcst_addr_offset + i ] );
        }
      }
    }
    else {
      for ( i = 0 ; i < a->dl_brdcst_addr_length ; ++i ) {
        if ( i != 0 ) printf( "." );
        printf( "%u", ((unsigned char*)a)[ a->dl_brdcst_addr_offset + i ] );
      }
    }
    printf( "\n" );
  }


  if ( a->dl_growth != 0 || a->dl_reserved != 0 ) {
    printf( "Warning: Reserved values are != 0\n" );
  }
}

ulong
DlpiMaxSdu( int dd )
{
  dl_u u;

  if ( GetDlpiInfo( dd, &u ) == 0 ) {
    return u.dl.info_ack.dl_max_sdu;
  }
  printf( "DlpiMaxSdu: Erro !!!\n" );
  return 0;
}

void
DlpiDisplayPhysAddr( int dd )
{
  struct strbuf ctrl;
  dl_u u;
  int Flags;
  int i;
  dl_phys_addr_ack_t *a;

  u.dl.physaddr_req.dl_primitive = DL_PHYS_ADDR_REQ;
  u.dl.physaddr_req.dl_addr_type = DL_CURR_PHYS_ADDR;
  ctrl.buf = (void*)&u.dl.physaddr_req;
  ctrl.maxlen = ctrl.len = sizeof u.dl.physaddr_req;
  putmsg( dd, &ctrl, NULL, RS_HIPRI );

  Flags = RS_HIPRI;
  ctrl.buf = (void*)&u;
  ctrl.len = ctrl.maxlen = sizeof u;
  if ( ( i = getmsg( dd, &ctrl, NULL, &Flags ) ) != 0 ) {
    perror( "getmsg" );
    printf( "i = %d\n", i );
    printf( "Prim = 0x%02lX\n", u.dl.dl_primitive );
    InternalError();
```



```
    exit( 1 );
  }
  if ( u.dl.physaddr_ack.dl_primitive != DL_PHYS_ADDR_ACK ) {
    printf( "Erro no DL_PHYS_ADDR_REQ: Primitiva: 0x%02lX\n", u.dl.dl_primitive
);
    if ( u.dl.dl_primitive == DL_ERROR_ACK ) {
      ShowErrorAck( &u.dl.error_ack );
    }
    return;
  }
  a = &u.dl.physaddr_ack;
  printf( "Interface Addr: " );
    for ( i = 0 ; i < 6 ; ++i ) {
      if ( i != 0 ) printf( ":" );
      printf( "%02X", ((unsigned char*)a)[ a->dl_addr_offset + i ] );
    }
    printf( "\n" );
}

void
ShowErrorAck( dl_error_ack_t *p )
{
  printf( "Primitive in error: 0x%02lX\n", p->dl_error_primitive );
  printf( "DLPI Error Code: 0x%02lX\n", p->dl_errno );
  printf( "UNIX Error Code: 0x%02lX -> %s\n", p->dl_unix_errno,
        strerror( p->dl_unix_errno ) );
}

void
DlpiAttach( int dd, int Port )
{
  struct strbuf ctrl;
  dl_u u;
  int Flags;

  u.dl.attach_req.dl_primitive = DL_ATTACH_REQ;
  u.dl.attach_req.dl_ppa = Port;
  ctrl.len = sizeof u.dl.attach_req;
  ctrl.buf = (void*)&u;
  putmsg( dd, &ctrl, NULL, 0 );
  Flags = RS_HIPRI;
  ctrl.maxlen = sizeof u;
  if ( getmsg( dd, &ctrl, NULL, &Flags ) < 0 ) {
    perror( "getmsg" );
    InternalError();
  }
  if ( u.dl.dl_primitive != DL_OK_ACK ) {
    printf( "Erro no DL_ATTACH_REQ: Primitiva: 0x%02lX\n", u.dl.dl_primitive );
    if ( u.dl.dl_primitive == DL_ERROR_ACK ) {
      if ( u.dl.error_ack.dl_errno == DL_BADPPA )
      {
        printf( "BADPPA: Tentativa de attach em interface inexistente\n" );
      }
      else {
```

```
        ShowErrorAck( &u.dl.error_ack );
      }
    }
    exit( 1 );
  }
  if ( DlpiVerbose ) {
    printf( "Attached to PPA #%d\n", Port );
  }
}

void
DlpiDetach( int dd )
{
  struct strbuf ctrl;
  dl_u u;
  int Flags;

  u.dl.detach_req.dl_primitive = DL_DETACH_REQ;
  ctrl.len = sizeof u.dl.detach_req;
  ctrl.buf = (void*)&u;
  ctrl.maxlen = sizeof u;
  if ( putmsg( dd, &ctrl, NULL, 0 ) < 0 ) {
    perror( "putmsg" );
    InternalError();
  }
  Flags = RS_HIPRI;
  if ( getmsg( dd, &ctrl, NULL, &Flags ) < 0 ) {
    perror( "getmsg" );
    InternalError();
  }
  if ( u.dl.dl_primitive != DL_OK_ACK ) {
    printf( "Erro no DL_DETACH_REQ: Primitiva: 0x%02lX\n", u.dl.dl_primitive );
    if ( u.dl.dl_primitive == DL_ERROR_ACK ) {
      ShowErrorAck( &u.dl.error_ack );
    }
    exit( 1 );
  }
  if ( DlpiVerbose ) {
    printf( "Detached\n" );
  }
}

void
DlpiBind( int dd, long sap )
{
  struct strbuf ctrl;
  dl_u u;
  int Flags;
  int i, len;
  unsigned char *p, *q;

  u.dl.bind_req.dl_primitive = DL_BIND_REQ;
  u.dl.bind_req.dl_sap = sap;
  u.dl.bind_req.dl_service_mode = DL_CLDLS;
```



```
  u.dl.bind_req.dl_xidtest_flg = 0;
  ctrl.len = sizeof u.dl.bind_req;
  ctrl.buf = (void*)&u;
  putmsg( dd, &ctrl, NULL, 0 );
  Flags = RS_HIPRI;
  if ( getmsg( dd, &ctrl, NULL, &Flags ) < 0 ) {
    perror( "getmsg" );
    InternalError();
  }
  if ( u.dl.dl_primitive != DL_BIND_ACK ) {
    if ( u.dl.dl_primitive == DL_ERROR_ACK ) {
      ShowErrorAck( (void*)&u );
    }
    printf( "Erro no DL_BIND_REQ: Primitiva: 0x%02lX\n", u.dl.dl_primitive );
    exit( 1 );
  }
  if ( DlpiVerbose ) {
    printf( "Bound to 0x%04lX\n", u.dl.bind_ack.dl_sap );
  }
  len = u.dl.bind_ack.dl_addr_length;
  if ( len < 6 ) {
    printf( "Endereco do BIND menor que 6...\n" );
    InternalError();
  }
  else {
    len -= ( i = 6 );
  }
  p = ((char*)&u) + u.dl.bind_ack.dl_addr_offset;
  q = (char *)&DlpiMyAddr;
  while ( i-- > 0 ) {
    *q++ = *p++;
  }
  if ( DlpiVerbose ) {
    printf( "Addr: %s", ether_ntoa( &DlpiMyAddr ) );
    while ( len-- > 0 ) {
      printf( ".%d", *p++ );
    }
    printf( "\n" );
  }
}

void
DlpiUnbind( int dd )
{
  struct strbuf ctrl;
  dl_u u;
  int Flags;

  u.dl.unbind_req.dl_primitive = DL_UNBIND_REQ;
  ctrl.len = sizeof u.dl.unbind_req;
  ctrl.buf = (void*)&u;
  ctrl.maxlen = sizeof u;
  if ( putmsg( dd, &ctrl, NULL, 0 ) < 0 ) {
    perror( "putmsg" );
```

```
    InternalError();
  }
  Flags = RS_HIPRI;
  if ( getmsg( dd, &ctrl, NULL, &Flags ) < 0 ) {
    perror( "getmsg" );
    InternalError();
  }
  if ( u.dl.dl_primitive != DL_OK_ACK ) {
    if ( u.dl.dl_primitive == DL_ERROR_ACK ) {
      ShowErrorAck( (void*)&u );
    }
    printf( "Erro no DL_UNBIND_REQ: Primitiva: 0x%02lX\n", u.dl.dl_primitive );
    exit( 1 );
  }
  if ( DlpiVerbose ) {
    printf( "Unbound\n" );
  }
}

void
DlpiPromiscuous( int dd, int Mode )
{
  struct strbuf ctrl;
  dl_u u;
  int Flags;

  u.dl.promiscon_req.dl_primitive = DL_PROMISCON_REQ;
  u.dl.promiscon_req.dl_level = Mode;
  ctrl.len = sizeof u.dl.promiscon_req;
  ctrl.buf = (void*)&u;
  putmsg( dd, &ctrl, NULL, 0 );
  Flags = 1;
  if ( getmsg( dd, &ctrl, NULL, &Flags ) < 0 ) {
    perror( "getmsg" );
    InternalError();
  }
  if ( u.dl.dl_primitive != DL_OK_ACK ) {
    if ( u.dl.dl_primitive == DL_ERROR_ACK ) {
      ShowErrorAck( (void*)&u );
    }
    printf( "Erro no DL_PROMISCON_REQ: Primitiva: 0x%02lX\n", u.dl.dl_primitive
);
    exit( 1 );
  }
  if ( u.dl.ok_ack.dl_correct_primitive != DL_PROMISCON_REQ ) {
    InternalError();
  }
  if ( DlpiVerbose ) {
    printf( "Promiscuous mode %d started...\n", Mode );
  }
}

int
DlpiRecvPacket( int dd, void *buffer, int buflen )
```



```
{
  struct strbuf ctrl, data;
  dl_u u;
  int Flags;

  ctrl.buf = (void *)&u;
  ctrl.maxlen = sizeof u;
  data.buf = ((char*)buffer)+14;
  data.maxlen = buflen;
  Flags = 0;
  if ( getmsg( dd, &ctrl, &data, &Flags ) < 0 ) {
    if ( errno == EAGAIN ) {
      /* Nao tem nada pronto para ser lido, retornar...                 */
      return 0;
    }
    perror( "getmsg" );
    InternalError();
  }
  if ( u.dl.dl_primitive != DL_UNITDATA_IND ) {
    if ( u.dl.dl_primitive == DL_ERROR_ACK ) {
      ShowErrorAck( (void*)&u );
    }
    printf( "Erro DlpiRecvPacket: Primitiva: 0x%02lX\n", u.dl.dl_primitive );
    exit( 1 );
  }
  memcpy( buffer, (char*)&u.dl.unitdata_ind + u.dl.unitdata_ind.dl_dest_addr_off
set, 6 );
  memcpy( ((char*)buffer)+6, (char*)&u.dl.unitdata_ind + u.dl.unitdata_ind.dl_sr
c_addr_offset, 8 );
  return 14+data.len;
}


const char *
DlpiServiceModeName( unsigned long Mode )
{
  static char Aux[80];

  switch ( Mode ) {
    case DL_CODLS:
      return "DL_CODLS";
    case DL_CLDLS:
      return "DL_CLDLS";
    case DL_CODLS|DL_CLDLS:
      return "DL_CODLS|DL_CLDLS";
    case DL_ACLDLS:
      return "DL_ACLDLS";
    default:
      sprintf( Aux, "UNKNOWN (0x%08lX)", Mode );
      return Aux;
  }
}
```

```
#ifndef _gtallc1_h
#define _gtallc1_h

/*************************************************************************/

typedef struct llc_pci_tipo1_t {
  /*  Sera' que eu poderia usar as definicoes do proprio Solaris ???    */
  /*  Onde tem ???                                                      */
  u_char DSap;
  u_char SSap;
  u_char Cmd;
} llc_pci_tipo1_t;

typedef struct snap_pci_t {
  u_char Org[3];                /*  0 for ether compatibility           */
  u_char Code[2];               /*  Ether Type                          */
} snap_pci_t;

typedef struct llc1_llc_addr_t {
  struct ether_addr     ether;
  u_char                sap;
} llc1_llc_addr_t;

typedef struct llc1_unitdata_ind_t {
  /*  Para ficar do mesmo tamanho do le_unitdata_ind_t, usei uns stuff. */
  dl_unitdata_ind_t dl;
  llc1_llc_addr_t dst;
  u_char stuff1;
  llc1_llc_addr_t src;
  u_char stuff2;
} llc1_unitdata_ind_t;

typedef struct llc1_unitdata_req_t {
  dl_unitdata_req_t dl;
  llc1_llc_addr_t dst;
} llc1_unitdata_req_t;

typedef struct llc1_uderror_ind_t {
  dl_uderror_ind_t dl;
  llc1_llc_addr_t  addr;
} llc1_uderror_ind_t;

typedef struct llc1_bind_ack_t {
  dl_bind_ack_t   dl;
  llc1_llc_addr_t addr;
} llc1_bind_ack_t;

typedef struct llc1_phys_addr_ack_t {
  dl_phys_addr_ack_t dl;
  struct {
    struct ether_addr  ether;
  } addr;
} llc1_phys_addr_ack_t;
```



```
typedef struct llc1_info_ack_t {
  dl_info_ack_t     dl;
  llc1_llc_addr_t   addr;
  char Pad[1];                  /*  Alinhamento de memoria  */
  struct ether_addr bcast;
} llc1_info_ack_t;

typedef union llc1_u {
  ulong                 dl_primitive;
  llc1_info_ack_t       info_ack;
  llc1_bind_ack_t       bind_ack;
  llc1_uderror_ind_t    uderror_ind;
  llc1_unitdata_req_t   unitdata_req;
  llc1_unitdata_ind_t   unitdata_ind;
} llc1_u;

#endif
```

lib/llc.h


```
#ifndef GTA_LLC_H
#define GTA_LLC_H

/*  Comandos de controle para a LLC tipo 1                                  */
#define LLC_UI          (0x03)          /*  Unnumbered Information     */
#define LLC_XID         (0xAF)          /*  eXchange IDentification    */
#define LLC_TEST        (0xE3)          /*  Test Command               */

/*  Definicao dos bits P/F ( Poll/Final )                                   */
/*  Estes bits devem ser OR'ed com os comandos, quando necessario.      */
#define LLC_PF          (0x10)
#define LLC_POLL        LLC_PF
#define LLC_FINAL       LLC_PF

#define LLC_COMMAND     (0x00)
#define LLC_RESPONSE    (0x01)
#define LLC_GROUP       (0x01)

/*  Definicao dos SAP's conhecidos                                          */
#define SAP_NULL        (0x00)  /*  Station LSAP Address                */
#define SAP_IMGT        (0x02)  /*  LLC Individual Management           */
#define SAP_GMGT        (0x03)  /*  LLC Group Management                */
#define SAP_SNA         (0x04)  /*  IBM SNA                             */
#define SAP_IP          (0x06)  /*  DOD TCP/IP                          */
#define SAP_GTANET      (0x94)  /*  Homenagem ao ano que passou         */
#define SAP_SNAP        (0xAA)  /*  Sub-Network Access Protocol         */
#define SAP_NOVELL      (0xE0)  /*  Novell Netware (IPX)                */
#define SAP_ISO8473     (0xFE)  /*  ISO CLNS IS 8473                    */
#define SAP_GLOBAL      (0xFF)  /*  Global DSAP                         */
#define SAP_UNBOUND     (-1L)   /*  Para o Cache: Unused.               */

#define SAP_IsMulti(s)  ((s)&LLC_GROUP)

#endif  /* GTA_LLC_H  */
```

lib/rtprio.c


```
#include <sys/types.h>
#include <sys/priocntl.h>
#include <sys/rtpriocntl.h>
#include <sys/tspriocntl.h>
#include <errno.h>

#include "rtprio.h"

int
SetTsPrio( int Prio )
{
  static id_t tscid = -1;
  pcinfo_t pcinfo;
  pcparms_t pcparms;

  if ( tscid == -1 ) {
    strncpy( pcinfo.pc_clname, "TS", PC_CLNMSZ );
    if ( priocntl( 0, 0, PC_GETCID, (caddr_t)&pcinfo ) == -1 ) {
      return -2;
    }
    tscid = pcinfo.pc_cid;
  }
  pcparms.pc_cid = tscid;
  ((tsparms_t*)(&pcparms.pc_clparms))->ts_upri     = Prio;
  ((tsparms_t*)(&pcparms.pc_clparms))->ts_uprilim  = Prio;
  if ( priocntl( P_PID, getpid(), PC_SETPARMS, (caddr_t)&pcparms ) == -1 ) {
    return -3;
  }

  return 0;
}

int
SetRtPrio( int Prio, float Quantum )
{
  static id_t rtcid = -1;
  pcinfo_t pcinfo;
  pcparms_t pcparms;
  long Secs, nSecs;

  if ( Quantum < 0 ) {
    errno = ERANGE;
    return -1;
  }
  if ( rtcid == -1 ) {
    strncpy( pcinfo.pc_clname, "RT", PC_CLNMSZ );
    if ( priocntl( 0, 0, PC_GETCID, (caddr_t)&pcinfo ) == -1 ) {
      return -2;
    }
    rtcid = pcinfo.pc_cid;
  }
  Secs = (long)Quantum;
  nSecs = (long)( (Quantum - Secs) * 1000000000L );
  pcparms.pc_cid = rtcid;
```

lib/rtprio.c


```
  ((rtparms_t*)(&pcparms.pc_clparms))->rt_pri     = Prio;
  ((rtparms_t*)(&pcparms.pc_clparms))->rt_tqsecs  = Secs;
  ((rtparms_t*)(&pcparms.pc_clparms))->rt_tqnsecs = nSecs;
  if ( priocntl( P_PID, getpid(), PC_SETPARMS, (caddr_t)&pcparms ) == -1 ) {
    return -3;
  }

  return 0;
}
```

medidas/Makefile


```
LIBDIR  = ../lib
CC      = gcc
OPT     =
CFLAGS= -pipe -Wall -g -I${LIBDIR}

libgta  = ${LIBDIR}/libgta.a

default:        all

all:            sender receiver sender-udp receiver-udp

clean:
                -rm -f receiver sender sender-udp receiver-udp
                -rm -f *.o core a.out

sender:         sender.c medidas.h ${libgta}
        -rm -f sender
        ${CC} ${CFLAGS} -o sender sender.c ${libgta} -lsocket

sender-udp:     sender-udp.c medidas.h ${libgta}
        -rm -f sender-udp
        ${CC} ${CFLAGS} -o sender-udp sender-udp.c ${libgta} -lsocket -lnsl

receiver:       receiver.c medidas.h
        -rm -f receiver
        ${CC} ${CFLAGS} -o receiver receiver.c ${libgta} -lsocket

receiver-udp:   receiver-udp.c medidas.h
        -rm -f receiver-udp
        ${CC} ${CFLAGS} -o receiver-udp receiver-udp.c ${libgta} -lsocket -lnsl

go:     go.c
        gcc -Wall -O -o go go.c
        chown 0.0 go
        chmod ug+s go
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/file.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <stropts.h>
#include <net/if.h>
#include  <netinet/in_systm.h>
#include <netinet/in.h>
#include <netinet/if_ether.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <netdb.h>
#include <fcntl.h>
#include <sys/dlpi.h>
#include <signal.h>
#include <sys/systeminfo.h>
#include <string.h>
#include <inet/led.h>
#include <sys/signal.h>
#include <sys/fault.h>
#include <sys/syscall.h>
#include <sys/procfs.h>
#include <sys/wait.h>

#include "dlpi.h"
#include "lib.h"
#include "gtallc1.h"
#include "llc.h"
#include "rtprio.h"

#include "medidas.h"

/*  Estes parametros podem ser mexidos à vontade durante as medidas     */

#define SEGUNDOS        ((ulong)10)     /* Tempo médio default p/ medida*/
#define DESTHOST        "buzios"        /* Destinatário dos pacotes     */
#define RETAGUARDA      1               /* Tempo de retaguarda em Seg.  */
#define QUANTUM         0.010           /* RtPrio quantum               */

#define BANDWIDTH       1250000L        /* larg. banda estimada (bytes) */
#define SMOOTH_CLOSE    1               /*  Unbind & Detach no close    */
#define MAGIC_HRTIME    0               /*  Usa hrtimer como Magic      */

#ifndef DEBUG
#define DEBUG           0
#endif

int ether_ntohost( char *hostname, struct ether_addr *e );
int ether_hostton( char *hostname, struct ether_addr *e );
```



```c
struct device_options_t {
  char *option;
  char *device;
  long ppa;
  long sap;
  long saplen;
  int  pcilen;          /* Tamanho dos dados adicionados ao pacote      */
} device_options[] = {
  { "le",       "/dev/le",      0, 0x4567, 2, 14 },
  { "llc",      "/dev/llc1mux", 0, 0x0010, 1, 17 },
  { "snap",     "/dev/gtasnap", 0, 0x4567, 2, 22 },
  { "lb",       "/dev/gtalb1a",-1, 0x4567, 2, 14 }, /* loopback sem ppa/sap  */
  { "lbllc",    "/dev/llc1mux", 1, 0x0010, 1, 17 },
  { "lbsnap",   "/dev/gtasnap", 1, 0x4567, 2, 22 },
  { "atm",      "/dev/fa",      0, 0x4567, 2, 0  }, /*  Não testado.        */
  { NULL,       NULL,           0, 0x0000, 0, 0  }
}, *device_option = &device_options[0];

struct mux_options_t {
  char *option;
  int   nproc;          /*  Número de processos (canais)                */
  int   preopen;        /*  Open antecipado ? ( 1 descritor p/ todos )  */
  int   pband1;         /*  Priority band da primeira instancia         */
  int   pband2;         /*  Priority band da segunda instancia          */
  int   pbands;         /*  Priority band somado p/ outras instancias   */
  int   rtprio1;        /*  Nivel de prioridade de Tempo Real           */
  int   rtprio2;        /*  Nivel de prioridade de Tempo Real           */
  int   rtprios;        /*  Nivel de prioridade de Tempo Real           */
  int   lbt;            /*  Limita por tempo, em vez de pacotes         */
} mux_options[] = {
  { "nomux",     1, 1,  0,  0,  0, 0, 0, 0, 1 },
  { "mux2",      2, 0,  0,  0,  0, 0, 0, 0, 1 }, /* Normal              */
  { "mux4",      4, 0,  0,  0,  0, 0, 0, 0, 1 },
  { "mux8",      8, 0,  0,  0,  0, 0, 0, 0, 1 },
  { "mux16",    16, 0,  0,  0,  0, 0, 0, 0, 1 },
  { "mux2p",     2, 1,  0,  0,  0, 0, 0, 0, 1 }, /* Single Channel      */
  { "mux4p",     4, 1,  0,  0,  0, 0, 0, 0, 1 },
  { "mux8p",     8, 1,  0,  0,  0, 0, 0, 0, 1 },
  { "mux16p",   16, 1,  0,  0,  0, 0, 0, 0, 1 },
  { "mux2b",     2, 0,  0,  2,  2, 0, 0, 0, 1 }, /*  Band Priority      */
  { "mux4b",     4, 0,  0,  2,  2, 0, 0, 0, 1 },
  { "mux8b",     8, 0,  0,  2,  2, 0, 0, 0, 1 },
  { "mux16b",   16, 0,  0,  2,  2, 0, 0, 0, 1 },
  { "mux2bp",    2, 1,  0,  2,  2, 0, 0, 0, 1 }, /*  SC & BP            */
  { "mux4bp",    4, 1,  0,  2,  2, 0, 0, 0, 1 },
  { "mux8bp",    8, 1,  0,  2,  2, 0, 0, 0, 1 },
  { "mux16bp",  16, 1,  0,  2,  2, 0, 0, 0, 1 },
  { "mux2r",     2, 0,  0,  0,  0, 1, 2, 1, 1 }, /*  RTPrio Variable    */
  { "mux4r",     4, 0,  0,  0,  0, 1, 2, 1, 1 },
  { "mux8r",     8, 0,  0,  0,  0, 1, 2, 1, 1 },
  { "mux16r",   16, 0,  0,  0,  0, 1, 2, 1, 1 },
  { "mux2rb",    2, 0,  0,  2,  2, 1, 1, 0, 1 }, /*  RTPrio Fix, BP Var */
  { "mux4rb",    4, 0,  0,  2,  2, 1, 1, 0, 1 },
```

```
  { "mux8rb",    8, 0,  0,  2,  2, 1, 1, 0, 1 },
  { "mux16rb",  16, 0,  0,  2,  2, 1, 1, 0, 1 },
  { "mux2br",    2, 0, 40, 30, -2, 1, 2, 1, 1 }, /*  BP & RTPrio, Inv */
  { "mux4br",    4, 0, 40, 30, -2, 1, 2, 1, 1 },
  { "mux8br",    8, 0, 40, 30, -2, 1, 2, 1, 1 },
  { "mux16br",  16, 0, 40, 30, -2, 1, 2, 1, 1 },
  { "mux2rx",    2, 0,  0,  0,  0, 1, 1, 0, 1 }, /*  RTPrio Fixed       */
  { "mux4rx",    4, 0,  0,  0,  0, 1, 1, 0, 1 },
  { "mux8rx",    8, 0,  0,  0,  0, 1, 1, 0, 1 },
  { "mux16rx",  16, 0,  0,  0,  0, 1, 1, 0, 1 },
  { NULL,        0, 0,  0,  0,  0, 0, 0, 0, 0 }  /*  Fim da lista       */
}, *mux_option = &mux_options[0];

#define MAX_RTPRIO      20              /*  Nenhum filho pode ter mais  */


#if 0
long Sizes[] = {   64,  128,  256,  384,  512,
                  640,  768,  896, 1024, 1152,
                 1280, 1408,    0,    0,    0,
                }; /*  0 -> Maximo suportado pelo driver, e terminador  */
#else
long Sizes[] = {   64,  128,  192,  256,  320,  384,
                  512,  640,  768, 1024, 1280,    0
                }; /*  0 -> Maximo suportado pelo driver, e terminador  */
#endif

char DestHost[80] = DESTHOST;

#define         PAYLOAD         ((ulong)(BANDWIDTH*Segundos))

int dd;
char hostname[80];
struct strbuf MsgCtrl, MsgData;
dl_u MsgCtrlBuffer;
char MsgDataBuffer[2048];
char DataFilename[80];
char *DataDir = DATA_DIR;
FILE *fp_data;
ulong MaxSdu;
hrtime_t Start, Stop, Elapsed;
ulong TmpVctx,   TmpIctx;
ulong StartVctx, StartIctx;
ulong TotalVctx, TotalIctx;
timestruc_t TmpCpu, StartCpu, TotalCpu;
float CpuPercent;
double Time, Pps;
long NPackets;
int PriorityBand;
volatile int SigUsr1Received, SigAlrmReceived;
int FailSafe = 0;
long Segundos = SEGUNDOS;
int Repeticoes = 1;
```



```
#define         mp      ((medidas_t*)MsgDataBuffer)
#define         NANO    1000000000L


int
OpenLog( void )
{
  if ( ( fp_data = fopen( DataFilename, "w" ) ) == NULL ) {
    printf( "Erro na abertura de %s\n", DataFilename );
    perror( "fopen" );
    return -1;
  }
  fprintf( fp_data, "# Arquivo de dados do emissor\n" );
  fprintf( fp_data, "# DstHost = %s, SrcHost = %s\n",
        DestHost, hostname );
  fprintf( fp_data, "%11s %7s %4s %8s %9s %6s %7s %7s %4s %2s %2s\n",
        "#    Mux-id", "Packets", "Size", "Time", "pps", "Mbps",
        "vctx", "ictx", "%cpu", "pb", "rt" );
  fflush( fp_data );
  return 0;
}

void
RegisterRates( ulong Magic, int MsgSize, int Instance, int RtPrio )
{
  double Mbps;

  Mbps = ( Pps * MsgSize * 8 ) / 1000000L;

  fprintf( fp_data,
        "%08lX %02d %7ld %4d %8.3f %9.3f %6.3f %7lu %7lu %4.1f %2d %2d\n",
        Magic, Instance, NPackets, MsgSize, Time, Pps, Mbps,
        TotalVctx, TotalIctx, CpuPercent*100, PriorityBand, RtPrio );

  fflush( fp_data );

  /*  Essa saida é apenas para feedback.  Os dados mesmo estão no arquivo  */
  printf(
        "%2d: %7ld Pcts, %8.3f s, %8.3f Pps, %7.4f Mbps, PBand %2d, RtPrio %2d\n
",
        Instance, NPackets, Time, Pps, Mbps, PriorityBand, RtPrio );
}

void
CloseLog( void )
{
  fclose( fp_data );
  sync();       /* Sim, eu sou paranoico !!!                            */
}


void
initdevice( void )
{
```

```
  DlpiVerbose = 0;
  if ( ( dd = open( device_option->device, O_RDWR ) ) < 0 ) {
    printf( "Erro: Nao pude acessar o dispositivo de rede.\n" );
    perror( device_option->device );
    exit( 1 );
  }

  if ( device_option->ppa != -1 ) {
    DlpiAttach( dd, device_option->ppa );
    DlpiBind( dd, device_option->sap );
  }

#if DEBUG
  DlpiDisplayFullInfo( dd );
#endif

  /*  Flush no device, para limpar lixo que tenha surgido ate' aqui.    */
  if ( ioctl( dd, I_FLUSH, (char *)FLUSHR ) < 0 ) perror( "flush" );

  MaxSdu = DlpiMaxSdu( dd );
  if ( mux_option->preopen ) {
    printf( "Dispositivo de rede completamente configurado\n" );
    printf( "Tamanho maximo da SDU: %lu\n", MaxSdu );
  }
}

void
cleardevice( void )
{
  #if SMOOTH_CLOSE
  if ( device_option->ppa != -1 ) {
    DlpiUnbind( dd );
    DlpiDetach( dd );
  }
  #endif
  close( dd );
  if ( mux_option->preopen ) {
    printf( "Dispositivo de rede fechado...\n" );
  }
}

void
InitData( long NPackets, int MsgSize, long Magic )
{
  MsgCtrlBuffer.dl.unitdata_req.dl_primitive = DL_UNITDATA_REQ;
  MsgCtrlBuffer.dl.unitdata_req.dl_dest_addr_length = 6 + device_option->saplen;
  MsgCtrlBuffer.dl.unitdata_req.dl_dest_addr_offset = sizeof(dl_unitdata_req_t);
  if (
    ether_hostton(
      DestHost,
      (struct ether_addr *)&MsgCtrlBuffer.data[ sizeof(dl_unitdata_req_t) ]
    ) != 0 )
  {
    printf( "Host desconhecido: %s\n", DestHost );
```



```
    InternalError();
  }

  switch ( device_option->saplen ) {
    case 1:
      *(u8*)&MsgCtrlBuffer.data[ 6 + sizeof(dl_unitdata_req_t) ] =
        device_option->sap;
      break;
    case 2:
      *(u16*)&MsgCtrlBuffer.data[ 6 + sizeof(dl_unitdata_req_t) ] =
        device_option->sap;
      break;
    default:
      InternalError();
  };

  mp->Size     = MsgSize;
  mp->RealSize = MsgSize - device_option->pcilen;
  mp->NPackets = NPackets;
  mp->Magic    = Magic;
  mp->Time     = Segundos;
  mp->Sequence = 0;
  mp->Instance = 0;
  strcpy( mp->Name, device_option->option );

  if ( mp->RealSize < sizeof *mp ) {
    InternalError();
  }

  MsgCtrl.buf  = (void *)&MsgCtrlBuffer;
  MsgCtrl.len  = sizeof(dl_unitdata_req_t) + 6 + device_option->saplen;
  MsgData.buf  = (void*)MsgDataBuffer;
  MsgData.len  = mp->RealSize;
}


inline int
SendOne( ulong n )
{
  mp->Sequence = n;
  if ( putpmsg( dd, &MsgCtrl, &MsgData, PriorityBand, MSG_BAND) < 0 ) {
    /*  XXX Se isso acontecer, a medida nao e' mais valida.  Abortar ??? */
    perror( "\nputmsg" );
    sleep( 1 );
    return 1;
  }
  return 0;
}

void
GetUsage( void )
{
  int fd;
  char fname[80];
```

```
  prusage_t pu;

  sprintf( fname, "/proc/%ld", getpid() );
  if ( ( fd = open( fname, O_RDONLY ) ) < 0 ) {
    perror( fname );
    exit( 1 );
  }

  if ( ioctl( fd, PIOCUSAGE, &pu ) < 0 ) {
    perror( "ioctl" );
    exit( 1 );
  }

  TmpVctx = pu.pr_vctx;
  TmpIctx = pu.pr_ictx;

  TmpCpu.tv_sec  = pu.pr_utime.tv_sec + pu.pr_stime.tv_sec;
  TmpCpu.tv_nsec = pu.pr_utime.tv_nsec + pu.pr_stime.tv_nsec;
  if ( TmpCpu.tv_nsec >= NANO ) {
    TmpCpu.tv_sec++;
    TmpCpu.tv_nsec -= NANO;
  }

  close( fd );
}

inline void
StartTiming( void )
{
  GetUsage();
  StartVctx = TmpVctx;
  StartIctx = TmpIctx;
  StartCpu  = TmpCpu;
  Start = gethrtime();
}

inline void
StopTiming( void )
{
  hrtime_t TotalCpuTime;

  Stop = gethrtime();
  GetUsage();
  Elapsed = Stop - Start;
  TotalVctx = TmpVctx - StartVctx;
  TotalIctx = TmpIctx - StartIctx;
  TotalCpu.tv_sec  = TmpCpu.tv_sec  - StartCpu.tv_sec;
  TotalCpu.tv_nsec = TmpCpu.tv_nsec - StartCpu.tv_nsec;
  if ( TotalCpu.tv_nsec < 0 ) {
    TotalCpu.tv_sec--;
    TotalCpu.tv_nsec += NANO;
  }
  TotalCpuTime = (TotalCpu.tv_sec * (hrtime_t)NANO) + TotalCpu.tv_nsec;
  CpuPercent = TotalCpuTime / (float)Elapsed;
```



```
  Time = (double)Elapsed / NANO;
}

void
SigHandler( int Sig )
{
  switch ( Sig ) {
    case SIGUSR1:
      SigUsr1Received = 1;
      break;
    case SIGALRM:
      SigAlrmReceived = 1;
      break;
    default:
      printf( "\nErro, recebi um signal não esperado...\n" );
      break;
  }
}

void
DoOnce( int MsgSize )
{
  int n;
  long Magic;
  pid_t pid;
  int error;
  int rtprio;
  struct sigaction act;
  ulong ExpectedPackets;

  if ( MsgSize < 64 ) {
    /*  Por que esta’ acontecendo isso ???                               */
    InternalError();
  }

  #if MAGIC_HRTIME
  do {
    Magic = gethrtime();
  } while ( Magic == 0 );
  #else
  Magic = time( NULL );
  #endif

  if ( FailSafe ) {
    ExpectedPackets = ( MsgSize > 128 ? PAYLOAD / MsgSize : PAYLOAD / 128 ) * 2;
  }
  else {
    ExpectedPackets = ~0;    /* Valor maximo...                            */
  }
  if ( mux_option->lbt ) {
    NPackets = 0;
    printf( "\nEnviando: Segundos %ld, Size %4d, Magic 0x%08lX",
          Segundos, MsgSize, Magic );
  }
```

```
  else {
    /*  O numero de pacotes estimado é tal que permita um tempo medio
     *  total da medida aproximadamente igual para todas as medidas.
     *  Isso não leva em consideração medidas com mux.
     */
    NPackets = MsgSize > 256 ? PAYLOAD / MsgSize : PAYLOAD / 256;
    printf( "\nEnviando: Total %ld, Size %4d, Magic 0x%08lX",
          NPackets, MsgSize, Magic );
  }
  fflush( stdout );

  /*  Prepara para a sinalização de grupo                               */
  act.sa_handler = SigHandler;
  sigemptyset( &act.sa_mask );
  act.sa_flags = SA_RESTART;
  sigaction( SIGUSR1, &act, NULL );
  sigaction( SIGALRM, &act, NULL );     /*  Cuidado com o sleep         */

  /*  Cria as multiplas instâncias para enviar com multiplexação.       */
  PriorityBand = mux_option->pband1;
  rtprio = mux_option->rtprio1;
  for ( n = 0 ; n < mux_option->nproc ; ++n ) {
    if ( ( pid = fork() ) == 0 ) {
      /*  Filho...   */

      /* Posso mudar o device_option no filho, pois já fiz o fork()      */
      device_option->sap += n * SAP_SOMA;

      /*  Prepara a prioridade de tempo real                            */
      if ( rtprio != 0 ) {
        if ( SetRtPrio( rtprio, QUANTUM ) != 0 ) {
          perror( "SetRtPrio" );
          InternalError();
        }
      }

      if ( !mux_option->preopen ) {
        initdevice();
      }
      InitData( NPackets, MsgSize, Magic );
      mp->Instance = n+1;

      /*  Prepara para esperar o marcador de início.                    */
      SigUsr1Received = 0;
      pause();
      if ( SigUsr1Received == 0 ) {
        /*  Paranoia pura e simples.                                    */
        InternalError();
      }

      SigAlrmReceived = SigUsr1Received = 0;
      alarm( Segundos );
      StartTiming();
      if ( mux_option->lbt ) {
```



```
        for ( n = 1 ;
          n < ExpectedPackets && SigAlrmReceived == 0 && SigUsr1Received == 0 ;
          ++n )
        {
          SendOne( n );
        }
        if ( n == ExpectedPackets ) {
          printf( "Aviso: Instância %d parou por feature FAILSAFE\n",
            mp->Instance );
        }
        mp->NPackets = NPackets = n + 0;
        while ( n <= NPackets ) {
          SendOne( n++ );
        }
        StopTiming();
      }
      else {
        for ( n = 1 ; n <= NPackets ; ++n ) {
          SendOne( n );
        }
        StopTiming();
      }

      sleep( RETAGUARDA );
      SendOne( n );              /*  n == NPackets+1  */
      if ( !mux_option->preopen ) {
        cleardevice();
      }

      Pps = (double)NPackets / Time;
      RegisterRates( Magic, MsgSize, mp->Instance, rtprio );

      exit( 0 );
    }
    if ( pid == -1 ) {
      perror( "fork" );
    PaiAbortando:
      if ( setpgid( 0, 0 ) ) {
        perror( "setpgid" );
        InternalError();
      }
      kill( -getpgrp(), SIGHUP );
      InternalError();
    }
    PriorityBand = mux_option->pband2  + n * mux_option->pbands;
    rtprio       = mux_option->rtprio2 + n * mux_option->rtprios;
  }

#if 0
  if ( SetRtPrio( MAX_RTPRIO, QUANTUM ) != 0 ) {
    perror( "SetRtPrio" );
    goto PaiAbortando;
  }
  else {
```

```
    printf( "\nPai rodando com RtPrio maximo" );
    fflush( stdout );
  }
#endif


  /*  Ja disparei todos os filhos.  Espera um tempinho, e acorda todos  */
  sleep( 2 );
  sync();       /* Sim, eu sou paranoico !!!                            */
  printf( "\n" );  /* Go, Mifuni, go...                                  */
  sync();       /* Sim, eu sou paranoico !!!                            */
  sleep( 1 );
  StartTiming();
  kill( -getpgrp(), SIGUSR1 );
  sleep( 1 );

  SigAlrmReceived = 0;
  if ( mux_option->lbt ) {
    alarm( Segundos + 2 );  /*  Faixa dxe segurança pra timeout.        */
  }

  /*  Espera por todos os filhos.                                        */
  /*  Não gostei, mas tá funcionando, tá bom.                            */
  for ( n = 0 ; n <= mux_option->nproc ; ++n ) {
    pid = wait( &error );  /* XXX - Eca...  */
    if ( pid == -1 ) {
      switch ( errno ) {
        case ECHILD:
          /*  Último filho morreu, cai fora do loop.                     */
          break;
        case EINTR:
          if ( SigAlrmReceived ) {
            /*  Ai galera, acabou o tempo !!!                            */
            kill( -getpgrp(), SIGUSR1 );
          }
          ++n;
          continue;
        default:
          perror( "wait" );
          kill( -getpgrp(), SIGTERM );
          exit( 1 );
          break;
      }
    }
  }
  StopTiming();

  /*  Anota tambem os dados do processo pai, por curiosidade.           */
  NPackets = 0;
  Pps = 0;
  PriorityBand = 0;
  RegisterRates( Magic, 0, 0, 0 );
  /*  Obs: O tempo de retaguarda também é incluído aqui.                */
```



```
  /*  Volta a prioridade (quase) normal.                                     */
  if ( SetTsPrio( 0 ) != 0 ) {
    perror( "SetTsPrio" );
    InternalError();
  }

  printf( "\n" );
}

/*
 *  Como estamos lidando com drivers de nucleo em desenvolvimento, todo
 *  cuidado e’ pouco.  Por garantia, vamos forcar um sync() do sistema.
 *  Isso exige que o programa rode como root, mas nao deve ser o unico
 *  ponto a exigir isso.
 */

void
PreSync( void )
{
  printf( "Syncing: " );        /*  Paranoid mode                          */
  fflush( stdout );
  sync();
  printf( "." );
  fflush( stdout );
  sync();
  printf( "." );
  sync();
  printf( "." );
  sync();
  printf( ".\n" );
}

void
ProgramSetup( void )
{
  setvbuf( stdout, NULL, _IOLBF, BUFSIZ );
  sprintf( DataFilename,
        "%s/%s-%s.snd", DataDir,
        device_option->option,
        mux_option->option );
  sysinfo( SI_HOSTNAME, hostname, sizeof(hostname)-1 );
  if ( getpgrp() != getpid() ) {
    /*  Para consertar isto, talvez eu tenha que usar o setpgid()       */
    printf( "Este programa tem que rodar como dono do seu process group\n" );
    printf( "Tente a opção -b\n" );
    InternalError();
  }
}

void
SetDeviceOption( const char *arg )
{
  struct device_options_t *op = device_options;
```

```
  while ( op->option ) {
    if ( !strcmp( op->option, arg ) ) {
      device_option = op;
      return;
    }
    ++op;
  }
  printf( "Opção de dispositivo não reconhecida\n" );
  exit( 1 );
}

void
SetMuxOption( const char *arg )
{
  struct mux_options_t *op = mux_options;

  while ( op->option ) {
    if ( !strcmp( op->option, arg ) ) {
      mux_option = op;
      return;
    }
    ++op;
  }
  printf( "Opção de multiplexação não reconhecida\n" );
  exit( 1 );
}

int
main( int argc, char * const *argv )
{
  int Repeticao;
  int i;

  /*  Parsing do comando de linha.                                    */
  /*  Isso tá uma eca, mas TCP/IP de fazer melhor.                    */
  while ( ( i = getopt( argc, argv, "bfs:r:d:" ) ) != -1 ) {
    switch ( i ) {
      case 'd':
        DataDir = strdup( optarg );
        break;
      case 'r':
        Repeticoes = atol( optarg );
        if ( Repeticoes <= 0 || Repeticoes > 100 ) {
          printf( "Faixa de repetições fora do aceitável...\n" );
          exit( 1 );
        }
        break;
      case 's':
        Segundos = atol( optarg );
        if ( Segundos <= 0 || Segundos > 3600 ) {
          printf( "Faixa de tempo fora do aceitável...\n" );
          exit( 1 );
        }
        break;
```



```
      case 'b':
        /* Batch Mode, seto um novo pgroup.                                  */
        /* Cuidado: Com isso perco o Job Control do Terminal                 */
        if ( setpgid( 0, 0 ) ) {
          perror( "setpgid" );
          exit( 1 );
        }
        printf( "Modo Batch: controle de terminal pode não funcionar\n" );
        break;
      case 'f':
        FailSafe = 1;
        printf( "Habilitando modo FAILSAFE (limitação de pacotes)\n" );
        break;
      default:
        printf( "Opção não conhecida: -%c\n", i );
        exit( 0 );
        break;
    }
  }
  argc -= optind;
  argv += optind;
  if ( argc > 0 ) {
    SetDeviceOption( argv[0] );
  }
  if ( argc > 1 ) {
    SetMuxOption( argv[1] );
  }

  ProgramSetup();
  PreSync();

  printf( "Opção de dispositivo \"%s\"\n", device_option->option );
  printf( "Opção de multiplexação \"%s\"\n", mux_option->option );
  printf( "Host: %s, Destination: %s\n", hostname, DestHost );
  printf( "Número de repetições da medida: %d\n", Repeticoes );
  printf( "Tempo de cada medida: %ld segundos\n", Segundos );
  printf( "Data file: %s\n", DataFilename );

  if ( OpenLog() != 0 ) exit( 1 );

  sync();       /* Sim, eu sou paranoico !!!                                 */

  if ( mux_option->preopen ) {
    printf( "Inicializando driver de comunicação\n" );
    initdevice();
  }
  else {
    /*  Mesmo que seja para trabalhar sem o preopen, tenho que abrir    */
    /*  o device, para poder ler o MaxSdu.  Horrível, mas funciona.     */
    initdevice();
    cleardevice();
  }
```

```
  for ( Repeticao = 0 ; Repeticao < Repeticoes ; ++Repeticao ) {
    i = 0;
    sync();     /* Sim, eu sou paranoico !!!                          */
    do {
      /* DoOnce( Sizes[i] != 0 ? Sizes[i] : MaxSdu ); */
      DoOnce( Sizes[i] != 0 ? Sizes[i] : 1514 );
    } while ( Sizes[i++] != 0 );
    fprintf( fp_data, "\n" );
    fflush( fp_data );
  }
  printf( "\n" );

  CloseLog();
  if ( mux_option->preopen ) {
    cleardevice();
  }

  return 0;
}
```



```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/file.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <stropts.h>
#include <net/if.h>
#include  <netinet/in_systm.h>
#include <netinet/in.h>
#include <netinet/if_ether.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <netdb.h>
#include <fcntl.h>
#include <sys/dlpi.h>
#include <signal.h>
#include <sys/systeminfo.h>
#include <string.h>
#include <sys/signal.h>
#include <sys/fault.h>
#include <sys/syscall.h>
#include <sys/procfs.h>
#include <sys/wait.h>
#include <sys/priocntl.h>
#include <sys/rtpriocntl.h>

#include "dlpi.h"
#include "lib.h"
#include "gtallc1.h"
#include "llc.h"
#include "rtprio.h"

#include "medidas.h"

/* Opcoes de funcionamento:                                            */
#define SMOOTH_CLOSE    1       /*  Unbind & Detach antes do close      */
#define MAX_LOST_INFO   10240
#define USE_PRIOCNTL    1

#define         NANO 1000000000L

int ether_ntohost( char *hostname, struct ether_addr *e );
int ether_hostton( char *hostname, struct ether_addr *e );

struct device_options_t {
  char *option;
  char *device;
  long ppa;
  long sap;
  long saplen;
```

```c
} device_options[] = {
  { "le",       "/dev/le",      0, 0x4567, 2 },
  { "llc",      "/dev/llc1mux", 0, 0x0010, 1 },
  { "snap",     "/dev/gtasnap", 0, 0x4567, 2 },
  { "lb",       "/dev/gtalb1b",-1, 0x4567, 2 }, /* loopback, sem ppa nem sap */
  { "lbllc",    "/dev/llc1mux", 2, 0x0010, 1 },
  { "lbsnap",   "/dev/gtasnap", 2, 0x4567, 2 },
  { NULL,       NULL,           0, 0x0000, 0 }
}, *device_option = &device_options[0];

struct mux_options_t {
  char *option;
  int   nproc;          /*  Número de processos (canais)                */
} mux_options[] = {
  { "nomux",     1, },
  { "mux2",      2, },
  { "mux4",      4, },
  { "mux8",      8, },
  { "mux16",    16, },
  { NULL,        0, }
}, *mux_option = &mux_options[0];

struct LostInfo_t {
  ulong first;
  ulong range;
} LostInfo[MAX_LOST_INFO];
int LostIndex = 0;


int dd;
char hostname[80];
struct strbuf MsgCtrl, MsgData;
dl_u MsgCtrlBuffer;
char MsgDataBuffer[2048];
char DataFilename[80];
FILE *fp_data;
ulong MaxSdu;
int Fim;
ulong LostInfoSize;
hrtime_t Start, Stop;
ulong TmpVctx,   TmpIctx;
ulong StartVctx, StartIctx;
ulong TotalVctx, TotalIctx;
timestruc_t TmpCpu, StartCpu, TotalCpu;
float CpuPercent;
double Time, Pps;
int Instance;
pid_t Filhos[32];
int Master = 1;         /* Processo pai                         */
ulong NPackets;

#define         mp      ((medidas_t*)MsgDataBuffer)
```



```c
int
OpenLog( void )
{
  if ( ( fp_data = fopen( DataFilename, "w" ) ) == NULL ) {
    printf( "Erro na abertura de %s\n", DataFilename );
    perror( "fopen" );
    return -1;
  }
  fprintf( fp_data, "# Arquivo de dados de medidas do receptor\n" );
  fprintf( fp_data, "# Host = %s\n", hostname );
  fprintf( fp_data, "%11s %7s %4s %8s %9s %6s %7s %7s %4s %4s\n",
        "#    Mux-id", "Packets", "Size", "Time", "pps", "Mbps",
        "vctx", "ictx", "%cpu", "Lost" );
  fflush( fp_data );
  return 0;
}

void
RegisterRates( ulong Magic, int MsgSize, int Instance, int Lost )
{
  double Mbps;

  Mbps = ( Pps * MsgSize * 8 ) / 1000000L;

  fprintf( fp_data,
        "%08lX-%02d %7ld %4d %8.3f %9.3f %6.3f %7lu %7lu %4.1f %4d\n",
        Magic, Instance, NPackets, MsgSize, Time, Pps, Mbps,
        TotalVctx, TotalIctx, CpuPercent*100, Lost );

  fflush( fp_data );

  /*  Essa saida é apenas para feedback.  Os dados mesmo estão no arquivo  */
  printf( "%2d: %7ld Pcts, %8.3f s, %8.3f Pps, %7.4f Mbps\n",
        Instance, NPackets, Time, Pps, Mbps );
}

void
CloseLog( void )
{
  fclose( fp_data );
  sync();       /* Sim, eu sou paranoico !!!                            */
}


void
initdevice( void )
{
  DlpiVerbose = 0;
  if ( ( dd = open( device_option->device, O_RDWR ) ) < 0 ) {
    printf( "Erro: Nao pude acessar o dispositivo de rede.\n" );
    perror( device_option->device );
    exit( 1 );
  }
```

```
  if ( device_option->ppa != -1 ) {
    DlpiAttach( dd, device_option->ppa );
    DlpiBind( dd, device_option->sap );
  }

#if DEBUG
  DlpiDisplayFullInfo( dd );
#endif

  /*  Flush no device, para limpar lixo que tenha surgido ate' aqui.    */
  if ( ioctl( dd, I_FLUSH, (char *)FLUSHR ) < 0 ) perror( "flush" );

  /* Necessario para o receptor nao bloquear no getmsg...               */
  fcntl( dd, F_SETFL, O_NONBLOCK );
}

void
cleardevice( void )
{
  fcntl( dd, F_SETFL, 0 );
  #if SMOOTH_CLOSE
  if ( device_option->ppa != -1 ) {
    DlpiUnbind( dd );
    DlpiDetach( dd );
  }
  #endif
  close( dd );
}


/*
 *  Como estamos lidando com drivers de nucleo em desenvolvimento, todo
 *  cuidado e' pouco.  Por garantia, vamos forcar um sync() do sistema.
 *  Isso exige que o programa rode como root, mas nao deve ser o unico
 *  ponto a exigir isso.
 */

void
PreSync( void )
{
  printf( "Syncing: " );        /*  Paranoid mode                         */
  fflush( stdout );
  sync();
  printf( "." );
  fflush( stdout );
  sync();
  printf( "." );
  sync();
  printf( "." );
  sync();
  printf( ".\n" );
}

void
```



```
ProgramSetup( void )
{
  setvbuf( stdout, NULL, _IOLBF, BUFSIZ );
  sprintf( DataFilename, "%s/%s-%s.rcv", DATA_DIR,
        device_option->option,
        mux_option->option );
  sysinfo( SI_HOSTNAME, hostname, sizeof(hostname)-1 );
}

void
SetDeviceOption( const char *arg )
{
  struct device_options_t *op = device_options;

  while ( op->option ) {
    if ( !strcmp( op->option, arg ) ) {
      device_option = op;
      return;
    }
    ++op;
  }
  printf( "Opção de dispositivo não reconhecida\n" );
  exit( 1 );
}

void
SetMuxOption( const char *arg )
{
  struct mux_options_t *op = mux_options;

  while ( op->option ) {
    if ( !strcmp( op->option, arg ) ) {
      mux_option = op;
      return;
    }
    ++op;
  }
  printf( "Opção de multiplexação não reconhecida\n" );
  exit( 1 );
}

void
SigHandler( int Sig )
{
  if ( Master ) {
    printf( "\nCapturado signal tipo %d\n", Sig );
  }
  signal( Sig, SigHandler );
  Fim = 1;
}

hrtime_t Elapsed;
ulong Count;
int MsgSize;
```

```
inline void
CyclePrepare( void )
{
  LostIndex = 0;
  Count = 0;
  memset( LostInfo, 0, sizeof LostInfo );
}

inline void
CycleEnd( ulong Magic )
{
  int i;
  int Lost;

  Lost = NPackets - Count;
  RegisterRates( Magic, MsgSize, Instance, Lost );
  if ( Lost ) {
    printf( "Pacotes perdidos: %d\n", Lost );
  }
  if ( LostIndex != 0 ) {
    fprintf( fp_data, "# Perdidos: (%lu:%lu)",
        LostInfo[0].first, LostInfo[0].range );
    printf( "Range: (%lu:%lu)",
        LostInfo[0].first, LostInfo[0].range );
    Count = LostInfo[0].range;
    for ( i = 1 ; i < LostIndex ; ++i ) {
      printf( ", (%lu:%lu)", LostInfo[i].first, LostInfo[i].range );
      fprintf( fp_data, ", (%lu:%lu)", LostInfo[i].first, LostInfo[i].range );
      Count += LostInfo[i].range;
    }
    printf( "\n" );
    fprintf( fp_data, "\n" );
    if ( Count != Lost ) {
      printf( "Epa !  Contadores de erro não concordam entre si !!!\n" );
      //InternalError();
    }
  }
  CyclePrepare();
}

inline void
Add2Lost( ulong Expected, ulong Received )
{
  LostInfo[LostIndex].first = Expected;
  LostInfo[LostIndex].range = Received - Expected;
  ++LostIndex;
}

void
GetUsage( void )
{
  static int fd = -1;
  char fname[80];
```



```
  prusage_t pu;
  prstatus_t pr;

  if ( fd == -1 ) {
    sprintf( fname, "/proc/%ld", getpid() );
    if ( ( fd = open( fname, O_RDONLY ) ) < 0 ) {
      perror( fname );
      exit( 1 );
    }
  }

  if ( ioctl( fd, PIOCUSAGE, &pu ) < 0 ) {
    perror( "ioctl" );
    exit( 1 );
  }

  if ( ioctl( fd, PIOCSTATUS, &pr ) < 0 ) {
    perror( "ioctl" );
    exit( 1 );
  }

  TmpVctx = pu.pr_vctx;
  TmpIctx = pu.pr_ictx;

  TmpCpu.tv_sec  = pr.pr_utime.tv_sec + pr.pr_stime.tv_sec;
  TmpCpu.tv_nsec = pr.pr_utime.tv_nsec + pr.pr_stime.tv_nsec;
  if ( TmpCpu.tv_nsec >= NANO ) {
    TmpCpu.tv_sec++;
    TmpCpu.tv_nsec -= NANO;
  }
}

inline void
StartTiming( void )
{
  GetUsage();
  StartVctx = TmpVctx;
  StartIctx = TmpIctx;
  StartCpu  = TmpCpu;
  Start = gethrtime();
}

inline void
StopTiming( void )
{
  hrtime_t TotalCpuTime;

  Stop = gethrtime();
  GetUsage();
  Elapsed = Stop - Start;
  TotalVctx = TmpVctx - StartVctx;
  TotalIctx = TmpIctx - StartIctx;
  TotalCpu.tv_sec  = TmpCpu.tv_sec  - StartCpu.tv_sec;
  TotalCpu.tv_nsec = TmpCpu.tv_nsec - StartCpu.tv_nsec;
```

```
  if ( TotalCpu.tv_nsec < 0 ) {
    TotalCpu.tv_sec--;
    TotalCpu.tv_nsec += NANO;
  }
  TotalCpuTime = (TotalCpu.tv_sec * (hrtime_t)NANO) + TotalCpu.tv_nsec;
  CpuPercent = TotalCpuTime / (float)Elapsed;
  Time = (double)Elapsed / NANO;
  Pps = (double)NPackets / Time;
}


void
MainLoop( void )
{
  struct pollfd Fds[1];
  ulong Magic;
  ulong Next;
  int i;

  int Flags;

  Fds[0].events = POLLRDNORM | POLLIN | POLLPRI;
  Fds[0].fd = dd;
  MsgCtrl.buf = (void *)&MsgCtrlBuffer;
  MsgCtrl.maxlen = sizeof MsgCtrlBuffer;
  MsgData.buf = MsgDataBuffer;
  MsgData.maxlen = sizeof MsgDataBuffer;

  Fim = 0;
  CyclePrepare();

  while ( !Fim ) {
    i = poll( Fds, 1, INFTIM );
    if ( i < 0 ) {
      if ( errno == EINTR )
        continue;
      perror( "poll" );
      exit( 1 );
    }
    else if ( i == 0 ) {
      printf( "Humm, poll retornou sem dado para ser lido...\n" );
      continue;
    }
    if ( Fds[0].revents != 0 ) {
      /*
      Fico em loop para receber o maximo de pacotes enquanto houver prontos
      para receber.  Talvez uma melhor opcao fosse usar o modulo bufmod do
      STREAMS do Solaris.  A versao nova deixou o modulo muito mais pratico
      para se usar.  Nao vou usar por que isso poderia tornar a medida irreal.
      */
      while ( 1 ) {
        Flags = 0;

        if ( getmsg( dd, &MsgCtrl, &MsgData, &Flags ) < 0 ) {
```



```
          if ( errno == EAGAIN ) {
            /* Nada pronto para ler, voltar ao loop de poll()           */
            break;
          }
          perror( "getmsg" );
          InternalError();
        }
        if ( MsgCtrlBuffer.dl.dl_primitive != DL_UNITDATA_IND ) {
          if ( MsgCtrlBuffer.dl.dl_primitive == DL_ERROR_ACK ) {
            ShowErrorAck( (void*)&MsgCtrlBuffer );
          }
          printf( "Receiver Error: Primitiva: 0x%02lX\n",
                MsgCtrlBuffer.dl.dl_primitive );
          exit( 1 );
        }

        /**************************************************************\

        A cada pacote, tenho que checar a maquina de estados interna
        e externa.  Isso e' meio complicado, mas acho que torna o
        programa mais generico.

        Possibilidades:

        - Pacote fora de ordem (perdi alguns pacotes)
        - Ultimo pacote pode ter sido perdido
        - Pacote de retaguarda (ignorar caso o ultimo tenha sido visto)
          - Posso tentar medir o tempo de retaguarda ?
            Talvez isso seja util para medir o tempo de buffer de saída.
        - Pacote normal (posso tentar um fast detection ?)

        \**************************************************************/


        if ( mp->Sequence == 1 ) {
          /* Inicio de um novo ciclo...                                 */
          if ( Count != 0 ) {
            /*  O ciclo anterior nao terminou !!!                       */
            /*  Nem mesmo o pacote de retaguarda chegou ???             */
            StopTiming();
            printf( "Aviso: Ciclo nao terminado !\n ");
            NPackets = Count;
            CycleEnd( Magic );
          }
          StartTiming();
          Magic = mp->Magic;
          Count = Next = mp->Sequence + 1;
          NPackets = mp->NPackets;
          MsgSize = mp->Size;
          if ( NPackets == 0 ) {
            printf( "Recebendo Magic 0x%08lX-%02d: Size %4d, Time %d\n",
              Magic, Instance, mp->Size, mp->Time );
          }
          else {
            printf( "Recebendo Magic 0x%08lX-%02d: Size %4d, Total %ld\n",
```

```
              Magic, Instance, mp->Size, NPackets );
          }
          if ( mp->RealSize != MsgData.len ) {
            printf( "Warning, wrong size: %d (%d)\n", MsgData.len, mp->RealSize
);
          }
          continue;
        }

        /*  Nao é o inicio de um novo ciclo, então deve ser continuação.  */

        if ( Count == 0 ) {
          /* Pode ser um pacote de retaguarda que não foi necessário.   */
          if ( mp->Sequence == ( mp->NPackets + 1 ) ) {
            continue;
          }
          /* Opa...  Perdi alguma coisa...                              */
          printf( "Pacote recebido (0x%08lX:%ld), ciclo não inicializado...\n",
                  mp->Magic,
                  mp->Sequence );
          /*  O que fazer nessa situacao ???                            */
          /*  Será que eu perdi o pacote 0, por conta de alguma         */
          /*  Multiplexação ???                                         */
          continue;
        }

        /*  Será que é o fim do ciclo ?                                 */
        if ( ( mp->NPackets != 0 ) && ( mp->Sequence >= mp->NPackets ) ) {
          NPackets = mp->NPackets;
          StopTiming();
          if ( mp->Sequence == mp->NPackets ) {
            /*  Fim da sequência, sem perda do último pacote.           */
          } else if ( mp->Sequence == ( mp->NPackets + 1 ) ) {
            /*  Perdi o último pacote, mas peguei a retaguarda.         */
            Add2Lost( Next, mp->Sequence-1 );
            printf( "Ciclo 0x%08lX-%02X: Salvo na retaguarda.\n", Magic, mp->Ins
tance );
          } else {
            /*  Epa, isso não deveria acontecer !!!                     */
            printf( "Erro: Número de sequência inválido: 0x%08lX:%ld\n",
              mp->Magic, mp->Sequence );
            CyclePrepare();
            continue;
          }

          /* Fim de um ciclo...                                         */
          CycleEnd( Magic );
          continue;
        }
        /*  Default: Nao faz nada ?  */

        if ( mp->Sequence != Next ) {
          Add2Lost( Next, mp->Sequence );
        }
```



```
          /*  Como fazer um FastPath para esse caso ???                 */
          ++Count;
          Next = mp->Sequence + 1;
        }
      }
    }
}


int
main( int argc, const char **argv )
{
  int n, error;
  pid_t pid;


  /*  Parsing do comando de linha.                                      */
  /*  Isso tá uma eca, mas TCP/IP de fazer melhor.                      */
  if ( argc > 1 ) {
    SetDeviceOption( argv[1] );
  }
  if ( argc > 2 ) {
    SetMuxOption( argv[2] );
  }
  ProgramSetup();
  PreSync();

  printf( "Opção de dispositivo \"%s\"\n", device_option->option );
  printf( "Opção de multiplexação \"%s\"\n", mux_option->option );
  printf( "Hostname: %s\n", hostname );
  printf( "Data file: %s\n", DataFilename );

  if ( OpenLog() != 0 ) exit( 1 );

  #if USE_PRIOCNTL
  if ( SetRtPrio( 5, 0.100 ) != 0 ) {
    perror( "SetRtPrio" );
    InternalError();
  }
  printf( "Operando em prioridade de tempo real\n" );
  #endif

  printf( "Inicializando os processos filhos: " );
  fflush( stdout );

  /*  Abre as multiplas instâncias para enviar com multiplexação.       */
  for ( n = 0 ; n < mux_option->nproc ; ++n ) {
    if ( ( pid = fork() ) == 0 ) {
      /*  Filho...   */

      /* Posso mudar o device_option no filho, pois já fiz o fork()     */
      Master = 0;
      Instance = n+1;
      device_option->sap += n * SAP_SOMA;
```

```
      initdevice();
      signal( SIGINT,  SIG_IGN );
      signal( SIGUSR1, SigHandler );
      sleep( 1 );               /*  Sincroniza com os irmaos            */
      MainLoop();
      cleardevice();
      exit( 0 );
    }
    if ( pid == -1 ) {
      perror( "fork" );
      exit( 1 );
      /*  Vou deixar um bando de filho rodando, não posso fazer nada.   */
      /*  Talvez, tentar montar um session id e matar todo mundo ?      */
      /*  Nao gostei.  Mudando o session Id eu perco o controle via     */
      /*  terminal.                                                     */
    }
    Filhos[n] = pid;
    printf( "." );
    fflush( stdout );
  }
  printf( "\n" );

  /*  Espera o sinal para terminar o programa...  */
  signal( SIGINT, SigHandler );
  pause();

  for ( n = 0 ; n < mux_option->nproc ; ++n ) {
    kill( Filhos[n], SIGTERM );
  }

  /*  Espera por todos os filhos.                                       */
  /*  Não gostei, mas tá funcionando, tá bom.                           */
  for ( n = 0 ; n <= mux_option->nproc ; ++n ) {
    error = wait( &error );  /* XXX - Eca...  */
  }
  if ( error != -1 || errno != ECHILD ) {
    printf( "Opa...\n" );
  }

  CloseLog();

  return 0;
}
```



```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/file.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <net/if.h>
#include  <netinet/in_systm.h>
#include <netinet/in.h>
#include <netinet/if_ether.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <netinet/udp.h>
#include <netdb.h>
#include <fcntl.h>
#include <sys/dlpi.h>
#include <signal.h>
#include <sys/systeminfo.h>
#include <string.h>
#include <inet/led.h>
#include <sys/signal.h>
#include <sys/fault.h>
#include <sys/syscall.h>
#include <sys/procfs.h>
#include <sys/wait.h>

#include "lib.h"
#include "rtprio.h"

#include "medidas.h"

/*  Estes parametros podem ser mexidos à vontade durante as medidas     */

#define SEGUNDOS        ((ulong)10)     /* Tempo médio default p/ medida*/
#define DESTHOST        "buzios.snap"   /* Destinatário dos pacotes     */
#define RETAGUARDA      1               /* Tempo de retaguarda em Seg.  */
#define QUANTUM         0.010           /* RtPrio quantum               */
#define BANDWIDTH       1250000L        /* larg. banda estimada (bytes) */
#define SMOOTH_CLOSE    1               /*  Unbind & Detach no close    */
#define MAGIC_HRTIME    0               /*  Usa hrtimer como Magic      */

#ifndef DEBUG
#define DEBUG           0
#endif

int ether_ntohost( char *hostname, struct ether_addr *e );
int ether_hostton( char *hostname, struct ether_addr *e );

struct mux_options_t {
  char *option;
  int   nproc;          /*  Número de processos (canais)                */
```

```
  int   preopen;        /*  Open antecipado ? ( 1 descritor p/ todos )  */
  int   pband1;         /*  Priority band da primeira instancia         */
  int   pband2;         /*  Priority band da segunda instancia          */
  int   pbands;         /*  Priority band somado p/ outras instancias   */
  int   rtprio1;        /*  Nivel de prioridade de Tempo Real           */
  int   rtprio2;        /*  Nivel de prioridade de Tempo Real           */
  int   rtprios;        /*  Nivel de prioridade de Tempo Real           */
  int   lbt;            /*  Limita por tempo, em vez de pacotes         */
} mux_options[] = {
  { "nomux",     1, 1,  0,  0,  0, 0, 0, 0, 1 },
  { "mux2",      2, 0,  0,  0,  0, 0, 0, 0, 1 }, /* Normal              */
  { "mux4",      4, 0,  0,  0,  0, 0, 0, 0, 1 },
  { "mux8",      8, 0,  0,  0,  0, 0, 0, 0, 1 },
  { "mux16",    16, 0,  0,  0,  0, 0, 0, 0, 1 },
  { "mux2p",     2, 1,  0,  0,  0, 0, 0, 0, 1 }, /* Single Channel      */
  { "mux4p",     4, 1,  0,  0,  0, 0, 0, 0, 1 },
  { "mux8p",     8, 1,  0,  0,  0, 0, 0, 0, 1 },
  { "mux16p",   16, 1,  0,  0,  0, 0, 0, 0, 1 },
  { "mux2r",     2, 0,  0,  0,  0, 1, 2, 1, 1 }, /*  RTPrio Variable    */
  { "mux4r",     4, 0,  0,  0,  0, 1, 2, 1, 1 },
  { "mux8r",     8, 0,  0,  0,  0, 1, 2, 1, 1 },
  { "mux16r",   16, 0,  0,  0,  0, 1, 2, 1, 1 },
  { "mux2rx",    2, 0,  0,  0,  0, 1, 1, 0, 1 }, /*  RTPrio Fixed       */
  { "mux4rx",    4, 0,  0,  0,  0, 1, 1, 0, 1 },
  { "mux8rx",    8, 0,  0,  0,  0, 1, 1, 0, 1 },
  { "mux16rx",  16, 0,  0,  0,  0, 1, 1, 0, 1 },
  { NULL,        0, 0,  0,  0,  0, 0, 0, 0, 0 }  /*  Fim da lista       */
}, *mux_option = &mux_options[0];

#define MAX_RTPRIO      20              /*  Nenhum filho pode ter mais  */

long Sizes[] = {  128,  192,  256,  320,  384,
                  512,  640,  768, 1024, 1280,    0
                }; /*  0 -> Maximo suportado pelo driver, e terminador  */

char DestHost[80] = DESTHOST;
struct sockaddr_in DestAddr;

#define         PAYLOAD         ((ulong)(BANDWIDTH*Segundos))

int dd;
char hostname[80];
char MsgDataBuffer[2048];
char DataFilename[80];
char *DataDir = DATA_DIR;
FILE *fp_data;
hrtime_t Start, Stop, Elapsed;
ulong TmpVctx,   TmpIctx;
ulong StartVctx, StartIctx;
ulong TotalVctx, TotalIctx;
timestruc_t TmpCpu, StartCpu, TotalCpu;
float CpuPercent;
double Time, Pps;
long NPackets;
```



```
int PriorityBand;
volatile int SigUsr1Received, SigAlrmReceived;
int FailSafe = 0;                               /*  Nao acredito no SIGALRM     */
long Segundos = SEGUNDOS;
int Repeticoes = 1;
int Sap = 40000;

#define         mp      ((medidas_t*)MsgDataBuffer)
#define         NANO    1000000000L

int
OpenLog( void )
{
  if ( ( fp_data = fopen( DataFilename, "w" ) ) == NULL ) {
    printf( "Erro na abertura de %s\n", DataFilename );
    perror( "fopen" );
    return -1;
  }
  fprintf( fp_data, "# Arquivo de dados do emissor\n" );
  fprintf( fp_data, "# DstHost = %s, SrcHost = %s\n",
        DestHost, hostname );
  fprintf( fp_data, "%11s %7s %4s %8s %9s %6s %7s %7s %4s %2s %2s\n",
        "#    Mux-id", "Packets", "Size", "Time", "pps", "Mbps",
        "vctx", "ictx", "%cpu", "pb", "rt" );
  fflush( fp_data );
  return 0;
}

void
RegisterRates( ulong Magic, int MsgSize, int Instance, int RtPrio )
{
  double Mbps;

  Mbps = ( Pps * MsgSize * 8 ) / 1000000L;

  fprintf( fp_data,
        "%08lX %02d %7ld %4d %8.3f %9.3f %6.3f %7lu %7lu %4.1f %2d %2d\n",
        Magic, Instance, NPackets, MsgSize, Time, Pps, Mbps,
        TotalVctx, TotalIctx, CpuPercent*100, PriorityBand, RtPrio );

  fflush( fp_data );

  /*  Essa saida é apenas para feedback.  Os dados mesmo estão no arquivo  */
  printf(
        "%2d: %7ld Pcts, %8.3f s, %8.3f Pps, %7.4f Mbps, PBand %2d, RtPrio %2d\n
",
        Instance, NPackets, Time, Pps, Mbps, PriorityBand, RtPrio );
}

void
CloseLog( void )
{
  fclose( fp_data );
  sync();       /* Sim, eu sou paranoico !!!                            */
```

```
}


void
initdevice( void )
{
  struct sockaddr_in sa;

  if ( ( dd = socket( AF_INET, SOCK_DGRAM, 0 ) ) < 0 ) {
    printf( "Erro: Nao pude acessar o dispositivo de rede.\n" );
    perror( "socket" );
    exit( 1 );
  }

  memset( &sa, 0, sizeof sa );
  sa.sin_family = AF_INET;
  sa.sin_port = Sap;
  sa.sin_addr.s_addr = INADDR_ANY;

  if ( bind( dd, (struct sockaddr *)&sa, sizeof sa ) ) {
    perror( "bind" );
    exit( 1 );
  }


  if ( mux_option->preopen ) {
    printf( "Dispositivo de rede completamente configurado\n" );
  }
}

void
cleardevice( void )
{
  close( dd );
  if ( mux_option->preopen ) {
    printf( "Dispositivo de rede fechado...\n" );
  }
}

void
InitData( long NPackets, int MsgSize, long Magic )
{
  struct hostent *hp;
  int ttl;

  memset( &DestAddr, 0, sizeof DestAddr );
  if ( ( hp = gethostbyname( DestHost ) ) == NULL ) {
    perror( "gethostbyname" );
    exit( 1 );
  }
  memcpy( &DestAddr.sin_addr.s_addr, hp->h_addr_list[0], sizeof DestAddr.sin_add
r.s_addr );
  DestAddr.sin_family = AF_INET;
  DestAddr.sin_port = Sap;
```



```
  ttl = 16;
  if ( setsockopt( dd, IPPROTO_IP, IP_TTL, (char *)&ttl, sizeof(ttl)) < 0 ) {
    perror( "setsockopt(IP_TTL)" );
    exit( 1 );
  }

  ttl = 1;
  if ( setsockopt( dd, SOL_SOCKET, SO_REUSEADDR, (char *)&ttl, sizeof(ttl)) < 0
) {
    perror( "setsockopt(SO_REUSEADDR)" );
    exit( 1 );
  }

  mp->Size = MsgSize;
  mp->RealSize = MsgSize
               - sizeof (struct udphdr)
               - sizeof (struct ip)
               - sizeof (struct ether_header)
               - 8 /* SNAP + LLC */;
  mp->NPackets = NPackets;
  mp->Magic = Magic;
  mp->Time = Segundos;
  mp->Sequence = 0;
  mp->Instance = 0;
  strcpy( mp->Name, "udp" );

  if ( mp->RealSize < sizeof *mp ) {
    printf( "\nErro: Tamanho do pacote insuficiente para os dados\n" );
    printf( "Tamanho: %d (%d), Minimo: %d\n",
        mp->RealSize, mp->Size, sizeof *mp );
    kill( -getpgrp(), SIGABRT );
    InternalError();
  }
}


inline int
SendOne( ulong n )
{
  mp->Sequence = n;
  if ( sendto( dd, (void *)MsgDataBuffer, mp->RealSize, MSG_DONTROUTE,
        (struct sockaddr *)&DestAddr, sizeof DestAddr ) < 0
  ) {
    perror( "\nsend" );
    sleep( 1 );
    return 1;
  }
  return 0;
}

void
GetUsage( void )
{
```

```
  int fd;
  char fname[80];
  prusage_t pu;

  sprintf( fname, "/proc/%ld", getpid() );
  if ( ( fd = open( fname, O_RDONLY ) ) < 0 ) {
    perror( fname );
    exit( 1 );
  }

  if ( ioctl( fd, PIOCUSAGE, &pu ) < 0 ) {
    perror( "ioctl" );
    exit( 1 );
  }

  TmpVctx = pu.pr_vctx;
  TmpIctx = pu.pr_ictx;

  TmpCpu.tv_sec  = pu.pr_utime.tv_sec + pu.pr_stime.tv_sec;
  TmpCpu.tv_nsec = pu.pr_utime.tv_nsec + pu.pr_stime.tv_nsec;
  if ( TmpCpu.tv_nsec >= NANO ) {
    TmpCpu.tv_sec++;
    TmpCpu.tv_nsec -= NANO;
  }

  close( fd );
}

inline void
StartTiming( void )
{
  GetUsage();
  StartVctx = TmpVctx;
  StartIctx = TmpIctx;
  StartCpu  = TmpCpu;
  Start = gethrtime();
}

inline void
StopTiming( void )
{
  hrtime_t TotalCpuTime;

  Stop = gethrtime();
  GetUsage();
  Elapsed = Stop - Start;
  TotalVctx = TmpVctx - StartVctx;
  TotalIctx = TmpIctx - StartIctx;
  TotalCpu.tv_sec  = TmpCpu.tv_sec  - StartCpu.tv_sec;
  TotalCpu.tv_nsec = TmpCpu.tv_nsec - StartCpu.tv_nsec;
  if ( TotalCpu.tv_nsec < 0 ) {
    TotalCpu.tv_sec--;
    TotalCpu.tv_nsec += NANO;
  }
```



```
  TotalCpuTime = (TotalCpu.tv_sec * (hrtime_t)NANO) + TotalCpu.tv_nsec;
  CpuPercent = TotalCpuTime / (float)Elapsed;
  Time = (double)Elapsed / NANO;
}

void
SigHandler( int Sig )
{
  switch ( Sig ) {
    case SIGUSR1:
      SigUsr1Received = 1;
      break;
    case SIGALRM:
      SigAlrmReceived = 1;
      break;
    default:
      printf( "\nErro, recebi um signal não esperado...\n" );
      break;
  }
}

void
DoOnce( int MsgSize )
{
  int n;
  long Magic;
  pid_t pid;
  int error;
  int rtprio;
  struct sigaction act;
  ulong ExpectedPackets;

  if ( MsgSize < 64 ) {
    /*  Por que esta’ acontecendo isso ???                          */
    InternalError();
  }

  #if MAGIC_HRTIME
  do {
    Magic = gethrtime();
  } while ( Magic == 0 );
  #else
  Magic = time( NULL );
  #endif

  if ( FailSafe ) {
    ExpectedPackets = ( MsgSize > 128 ? PAYLOAD / MsgSize : PAYLOAD / 128 ) * 2;
  }
  else {
    ExpectedPackets = ~0;    /* Valor maximo...                          */
  }
  if ( mux_option->lbt ) {
    NPackets = 0;
    printf( "\nEnviando: Segundos %ld, Size %4d, Magic 0x%08lX",
```

```
        Segundos, MsgSize, Magic );
  }
  else {
    /*  O numero de pacotes estimado é tal que permita um tempo medio
     *  total da medida aproximadamente igual para todas as medidas.
     *  Isso não leva em consideração medidas com mux.
     */
    NPackets = MsgSize > 256 ? PAYLOAD / MsgSize : PAYLOAD / 256;
    printf( "\nEnviando: Total %ld, Size %4d, Magic 0x%08lX",
          NPackets, MsgSize, Magic );
  }
  fflush( stdout );

  /*  Prepara para a sinalização de grupo                               */
  act.sa_handler = SigHandler;
  sigemptyset( &act.sa_mask );
  act.sa_flags = SA_RESTART;
  sigaction( SIGUSR1, &act, NULL );
  sigaction( SIGALRM, &act, NULL );     /*  Cuidado com o sleep         */

  /*  Cria as multiplas instâncias para enviar com multiplexação.       */
  PriorityBand = mux_option->pband1;
  rtprio = mux_option->rtprio1;
  for ( n = 0 ; n < mux_option->nproc ; ++n ) {
    if ( ( pid = fork() ) == 0 ) {
      /*  Filho...   */

      Sap += n * SAP_SOMA;

      /*  Prepara a prioridade de tempo real                            */
      if ( rtprio != 0 ) {
        if ( SetRtPrio( rtprio, QUANTUM ) != 0 ) {
          perror( "SetRtPrio" );
          InternalError();
        }
      }

      if ( !mux_option->preopen ) {
        initdevice();
      }
      InitData( NPackets, MsgSize, Magic );
      mp->Instance = n+1;

      /*  Prepara para esperar o marcador de início.                    */
      SigUsr1Received = 0;
      pause();
      if ( SigUsr1Received == 0 ) {
        /*  Paranoia pura e simples.                                    */
        InternalError();
      }

      SigAlrmReceived = SigUsr1Received = 0;
      alarm( Segundos );
      StartTiming();
```



```
      if ( mux_option->lbt ) {
        for ( n = 1 ;
          n < ExpectedPackets && SigAlrmReceived == 0 && SigUsr1Received == 0 ;
          ++n )
        {
          SendOne( n );
        }
        if ( n == ExpectedPackets ) {
          printf( "Aviso: Instância %d parou por feature FAILSAFE\n",
            mp->Instance );
        }
        mp->NPackets = NPackets = n + 0;
        while ( n <= NPackets ) {
          SendOne( n++ );
        }
        StopTiming();
      }
      else {
        for ( n = 1 ; n <= NPackets ; ++n ) {
          SendOne( n );
        }
        StopTiming();
      }

      sleep( RETAGUARDA );
      SendOne( n );             /*  n == NPackets+1  */
      if ( !mux_option->preopen ) {
        cleardevice();
      }

      Pps = (double)NPackets / Time;
      RegisterRates( Magic, MsgSize, mp->Instance, rtprio );

      exit( 0 );
    }
    if ( pid == -1 ) {
      perror( "fork" );
    PaiAbortando:
      if ( setpgid( 0, 0 ) ) {
        perror( "setpgid" );
        InternalError();
      }
      kill( -getpgrp(), SIGHUP );
      InternalError();
    }
    PriorityBand = mux_option->pband2  + n * mux_option->pbands;
    rtprio       = mux_option->rtprio2 + n * mux_option->rtprios;
  }

#if 0
  if ( SetRtPrio( MAX_RTPRIO, QUANTUM ) != 0 ) {
    perror( "SetRtPrio" );
    goto PaiAbortando;
  }
```

```
  else {
    printf( "\nPai rodando com RtPrio maximo" );
    fflush( stdout );
  }
#endif


  /*  Ja disparei todos os filhos.  Espera um tempinho, e acorda todos  */
  sleep( 2 );
  sync();        /* Sim, eu sou paranoico !!!                          */
  printf( "\n" );  /* Go, Mifuni, go...                                */
  sync();        /* Sim, eu sou paranoico !!!                          */
  sleep( 1 );
  StartTiming();
  kill( -getpgrp(), SIGUSR1 );
  sleep( 1 );

  SigAlrmReceived = 0;
  if ( mux_option->lbt ) {
    alarm( Segundos + 2 );  /*  Faixa dxe segurança pra timeout.        */
  }

  /*  Espera por todos os filhos.                                       */
  /*  Não gostei, mas tá funcionando, tá bom.                           */
  for ( n = 0 ; n <= mux_option->nproc ; ++n ) {
    pid = wait( &error );  /* XXX - Eca...  */
    if ( pid == -1 ) {
      switch ( errno ) {
        case ECHILD:
          /*  Último filho morreu, cai fora do loop.                    */
          break;
        case EINTR:
          if ( SigAlrmReceived ) {
            /*  Ai galera, acabou o tempo !!!                           */
            kill( -getpgrp(), SIGUSR1 );
          }
          ++n;
          continue;
        default:
          perror( "wait" );
          kill( -getpgrp(), SIGTERM );
          exit( 1 );
          break;
      }
    }
  }
  StopTiming();

  /*  Anota tambem os dados do processo pai, por curiosidade.           */
  NPackets = 0;
  Pps = 0;
  PriorityBand = 0;
  RegisterRates( Magic, 0, 0, 0 );
  /*  Obs: O tempo de retaguarda também é incluído aqui.                */
```



```

  /*  Volta a prioridade (quase) normal.                                */
  if ( SetTsPrio( 0 ) != 0 ) {
    perror( "SetTsPrio" );
    InternalError();
  }

  printf( "\n" );
}

/*
 *  Como estamos lidando com drivers de nucleo em desenvolvimento, todo
 *  cuidado e’ pouco.  Por garantia, vamos forcar um sync() do sistema.
 *  Isso exige que o programa rode como root, mas nao deve ser o unico
 *  ponto a exigir isso.
 */

void
PreSync( void )
{
  printf( "Syncing: " );        /*  Paranoid mode                       */
  fflush( stdout );
  sync();
  printf( "." );
  fflush( stdout );
  sync();
  printf( "." );
  sync();
  printf( "." );
  sync();
  printf( ".\n" );
}

void
ProgramSetup( void )
{
  setvbuf( stdout, NULL, _IOLBF, BUFSIZ );
  sprintf( DataFilename,
        "%s/udp-%s-%s.snd", DataDir,
        DestHost,
        mux_option->option );
  sysinfo( SI_HOSTNAME, hostname, sizeof(hostname)-1 );
  if ( getpgrp() != getpid() ) {
    /*  Para consertar isto, talvez eu tenha que usar o setpgid()       */
    printf( "Este programa tem que rodar como dono do seu process group\n" );
    printf( "Tente a opção -b\n" );
    InternalError();
  }
}

void
SetMuxOption( const char *arg )
{
  struct mux_options_t *op = mux_options;
```

```
  while ( op->option ) {
    if ( !strcmp( op->option, arg ) ) {
      mux_option = op;
      return;
    }
    ++op;
  }
  printf( "Opção de multiplexação não reconhecida\n" );
  exit( 1 );
}

int
main( int argc, char * const *argv )
{
  int Repeticao;
  int i;

  /*  Parsing do comando de linha.                                       */
  /*  Isso tá uma eca, mas TCP/IP de fazer melhor.                       */
  while ( ( i = getopt( argc, argv, "bfs:r:d:" ) ) != -1 ) {
    switch ( i ) {
      case 'd':
        DataDir = strdup( optarg );
        break;
      case 'r':
        Repeticoes = atol( optarg );
        if ( Repeticoes <= 0 || Repeticoes > 100 ) {
          printf( "Faixa de repetições fora do aceitável...\n" );
          exit( 1 );
        }
        break;
      case 's':
        Segundos = atol( optarg );
        if ( Segundos <= 0 || Segundos > 3600 ) {
          printf( "Faixa de tempo fora do aceitável...\n" );
          exit( 1 );
        }
        break;
      case 'b':
        /* Batch Mode, seto um novo pgroup.                                */
        /* Cuidado: Com isso perco o Job Control do Terminal               */
        if ( setpgid( 0, 0 ) ) {
          perror( "setpgid" );
          exit( 1 );
        }
        printf( "Modo Batch: controle de terminal pode não funcionar\n" );
        break;
      case 'f':
        FailSafe = 1;
        printf( "Habilitando modo FAILSAFE (limitação de pacotes)\n" );
        break;
      default:
        printf( "Opção não conhecida: -%c\n", i );
```



```
        exit( 0 );
        break;
    }
  }
  argc -= optind;
  argv += optind;
  if ( argc > 0 ) {
    strcpy( DestHost, argv[0] );
  }
  if ( argc > 1 ) {
    SetMuxOption( argv[1] );
  }

  ProgramSetup();
  PreSync();

  printf( "Opção de dispositivo \"%s\"\n", "udp" );
  printf( "Opção de multiplexação \"%s\"\n", mux_option->option );
  printf( "Host: %s, Destination: %s\n", hostname, DestHost );
  printf( "Número de repetições da medida: %d\n", Repeticoes );
  printf( "Tempo de cada medida: %ld segundos\n", Segundos );
  printf( "Data file: %s\n", DataFilename );

  if ( OpenLog() != 0 ) exit( 1 );

  sync();       /* Sim, eu sou paranoico !!!                                  */

  if ( mux_option->preopen ) {
    printf( "Inicializando driver de comunicação\n" );
    initdevice();
  }
  else {
    initdevice();
    cleardevice();
  }

  for ( Repeticao = 0 ; Repeticao < Repeticoes ; ++Repeticao ) {
    i = 0;
    sync();     /* Sim, eu sou paranoico !!!                                  */
    do {
      DoOnce( Sizes[i] != 0 ? Sizes[i] : 1508 );
    } while ( Sizes[i++] != 0 );
    fprintf( fp_data, "\n" );
    fflush( fp_data );
  }
  printf( "\n" );

  CloseLog();
  if ( mux_option->preopen ) {
    cleardevice();
  }
  return 0;
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/file.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <stropts.h>
#include <net/if.h>
#include  <netinet/in_systm.h>
#include <netinet/in.h>
#include <netinet/if_ether.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <netinet/udp.h>
#include <netdb.h>
#include <fcntl.h>
#include <sys/dlpi.h>
#include <signal.h>
#include <sys/systeminfo.h>
#include <string.h>
#include <sys/signal.h>
#include <sys/fault.h>
#include <sys/syscall.h>
#include <sys/procfs.h>
#include <sys/wait.h>
#include <sys/priocntl.h>
#include <sys/rtpriocntl.h>

#include "lib.h"
#include "rtprio.h"

#include "medidas.h"

/* Opcoes de funcionamento:                                              */
#define SMOOTH_CLOSE    1       /*  Unbind & Detach antes do close      */
#define MAX_LOST_INFO   10240
#define USE_PRIOCNTL    1

#define         NANO 1000000000L

int ether_ntohost( char *hostname, struct ether_addr *e );
int ether_hostton( char *hostname, struct ether_addr *e );

struct mux_options_t {
  char *option;
  int   nproc;          /*  Número de processos (canais)              */
} mux_options[] = {
  { "nomux",     1, },
  { "mux2",      2, },
  { "mux4",      4, },
  { "mux8",      8, },
```



```
  { "mux16",    16, },
  { NULL,        0, }
}, *mux_option = &mux_options[0];

struct LostInfo_t {
  ulong first;
  ulong range;
} LostInfo[MAX_LOST_INFO];
int LostIndex = 0;


int dd;
char hostname[80];
char MsgDataBuffer[2048];
char DataFilename[80];
FILE *fp_data;
int Fim;
ulong LostInfoSize;
hrtime_t Start, Stop;
ulong TmpVctx,   TmpIctx;
ulong StartVctx, StartIctx;
ulong TotalVctx, TotalIctx;
timestruc_t TmpCpu, StartCpu, TotalCpu;
float CpuPercent;
double Time, Pps;
int Instance;
pid_t Filhos[32];
int Master = 1;         /* Processo pai                          */
ulong NPackets;
int Sap = 40000;

#define         mp      ((medidas_t*)MsgDataBuffer)


int
OpenLog( void )
{
  if ( ( fp_data = fopen( DataFilename, "w" ) ) == NULL ) {
    printf( "Erro na abertura de %s\n", DataFilename );
    perror( "fopen" );
    return -1;
  }
  fprintf( fp_data, "# Arquivo de dados de medidas do receptor\n" );
  fprintf( fp_data, "# Host = %s\n", hostname );
  fprintf( fp_data, "%11s %7s %4s %8s %9s %6s %7s %7s %4s %4s\n",
        "#    Mux-id", "Packets", "Size", "Time", "pps", "Mbps",
        "vctx", "ictx", "%cpu", "Lost" );
  fflush( fp_data );
  return 0;
}

void
RegisterRates( ulong Magic, int MsgSize, int Instance, int Lost )
{
```

```
  double Mbps;

  Mbps = ( Pps * MsgSize * 8 ) / 1000000L;

  fprintf( fp_data,
        "%08lX-%02d %7ld %4d %8.3f %9.3f %6.3f %7lu %7lu %4.1f %4d\n",
        Magic, Instance, NPackets, MsgSize, Time, Pps, Mbps,
        TotalVctx, TotalIctx, CpuPercent*100, Lost );

  fflush( fp_data );

  /*  Essa saida é apenas para feedback.  Os dados mesmo estão no arquivo  */
  printf( "%2d: %7ld Pcts, %8.3f s, %8.3f Pps, %7.4f Mbps\n",
        Instance, NPackets, Time, Pps, Mbps );
}

void
CloseLog( void )
{
  fclose( fp_data );
  sync();       /* Sim, eu sou paranoico !!!                            */
}


void
initdevice( void )
{
  struct sockaddr_in sa;


  if ( ( dd = socket( AF_INET, SOCK_DGRAM, 0 ) ) < 0 ) {
    printf( "Erro: Nao pude acessar o dispositivo de rede.\n" );
    perror( "socket" );
    exit( 1 );
  }

  memset( &sa, 0, sizeof sa );
  sa.sin_family = AF_INET;
  sa.sin_port = Sap;
  sa.sin_addr.s_addr = INADDR_ANY;

  if ( bind( dd, (struct sockaddr *)&sa, sizeof sa ) ) {
    perror( "bind" );
    exit( 1 );
  }

  /* Necessario para o receptor nao bloquear no getmsg...               */
  fcntl( dd, F_SETFL, O_NONBLOCK );
}

void
cleardevice( void )
{
  fcntl( dd, F_SETFL, 0 );
```



```
  close( dd );
}


/*
 *  Como estamos lidando com drivers de nucleo em desenvolvimento, todo
 *  cuidado e' pouco.  Por garantia, vamos forcar um sync() do sistema.
 *  Isso exige que o programa rode como root, mas nao deve ser o unico
 *  ponto a exigir isso.
 */

void
PreSync( void )
{
  printf( "Syncing: " );        /*  Paranoid mode                       */
  fflush( stdout );
  sync();
  printf( "." );
  fflush( stdout );
  sync();
  printf( "." );
  sync();
  printf( "." );
  sync();
  printf( ".\n" );
}

void
ProgramSetup( void )
{
  setvbuf( stdout, NULL, _IOLBF, BUFSIZ );
  sprintf( DataFilename, "%s/%s-%s.rcv", DATA_DIR,
        "udp",
        mux_option->option );
  sysinfo( SI_HOSTNAME, hostname, sizeof(hostname)-1 );
}

void
SetMuxOption( const char *arg )
{
  struct mux_options_t *op = mux_options;

  while ( op->option ) {
    if ( !strcmp( op->option, arg ) ) {
      mux_option = op;
      return;
    }
    ++op;
  }
  printf( "Opção de multiplexação não reconhecida\n" );
  exit( 1 );
}

void
```

```c
SigHandler( int Sig )
{
  if ( Master ) {
    printf( "\nCapturado signal tipo %d\n", Sig );
  }
  signal( Sig, SigHandler );
  Fim = 1;
}

hrtime_t Elapsed;
ulong Count;
int MsgSize;

inline void
CyclePrepare( void )
{
  LostIndex = 0;
  Count = 0;
  memset( LostInfo, 0, sizeof LostInfo );
}

inline void
CycleEnd( ulong Magic )
{
  int i;
  int Lost;

  Lost = NPackets - Count;
  RegisterRates( Magic, MsgSize, Instance, Lost );
  if ( Lost ) {
    printf( "Pacotes perdidos: %d\n", Lost );
  }
  if ( LostIndex != 0 ) {
    fprintf( fp_data, "# Perdidos: (%lu:%lu)",
        LostInfo[0].first, LostInfo[0].range );
    printf( "Range: (%lu:%lu)",
        LostInfo[0].first, LostInfo[0].range );
    Count = LostInfo[0].range;
    for ( i = 1 ; i < LostIndex ; ++i ) {
      printf( ", (%lu:%lu)", LostInfo[i].first, LostInfo[i].range );
      fprintf( fp_data, ", (%lu:%lu)", LostInfo[i].first, LostInfo[i].range );
      Count += LostInfo[i].range;
    }
    printf( "\n" );
    fprintf( fp_data, "\n" );
    if ( Count != Lost ) {
      printf( "Epa !  Contadores de erro não concordam entre si !!!\n" );
      //InternalError();
    }
  }
  CyclePrepare();
}

inline void
```



```c
Add2Lost( ulong Expected, ulong Received )
{
  LostInfo[LostIndex].first = Expected;
  LostInfo[LostIndex].range = Received - Expected;
  ++LostIndex;
}

void
GetUsage( void )
{
  static int fd = -1;
  char fname[80];
  prusage_t pu;
  prstatus_t pr;

  if ( fd == -1 ) {
    sprintf( fname, "/proc/%ld", getpid() );
    if ( ( fd = open( fname, O_RDONLY ) ) < 0 ) {
      perror( fname );
      exit( 1 );
    }
  }

  if ( ioctl( fd, PIOCUSAGE, &pu ) < 0 ) {
    perror( "ioctl" );
    exit( 1 );
  }

  if ( ioctl( fd, PIOCSTATUS, &pr ) < 0 ) {
    perror( "ioctl" );
    exit( 1 );
  }

  TmpVctx = pu.pr_vctx;
  TmpIctx = pu.pr_ictx;

  TmpCpu.tv_sec  = pr.pr_utime.tv_sec + pr.pr_stime.tv_sec;
  TmpCpu.tv_nsec = pr.pr_utime.tv_nsec + pr.pr_stime.tv_nsec;
  if ( TmpCpu.tv_nsec >= NANO ) {
    TmpCpu.tv_sec++;
    TmpCpu.tv_nsec -= NANO;
  }
}

inline void
StartTiming( void )
{
  GetUsage();
  StartVctx = TmpVctx;
  StartIctx = TmpIctx;
  StartCpu  = TmpCpu;
  Start = gethrtime();
}
```

```
inline void
StopTiming( void )
{
  hrtime_t TotalCpuTime;

  Stop = gethrtime();
  GetUsage();
  Elapsed = Stop - Start;
  TotalVctx = TmpVctx - StartVctx;
  TotalIctx = TmpIctx - StartIctx;
  TotalCpu.tv_sec  = TmpCpu.tv_sec  - StartCpu.tv_sec;
  TotalCpu.tv_nsec = TmpCpu.tv_nsec - StartCpu.tv_nsec;
  if ( TotalCpu.tv_nsec < 0 ) {
    TotalCpu.tv_sec--;
    TotalCpu.tv_nsec += NANO;
  }
  TotalCpuTime = (TotalCpu.tv_sec * (hrtime_t)NANO) + TotalCpu.tv_nsec;
  CpuPercent = TotalCpuTime / (float)Elapsed;
  Time = (double)Elapsed / NANO;
  Pps = (double)NPackets / Time;
}


void
MainLoop( void )
{
  struct pollfd Fds[1];
  ulong Magic;
  ulong Next;
  int i;
  int Flags, RecvLen;

  Fds[0].events = POLLRDNORM | POLLIN | POLLPRI;
  Fds[0].fd = dd;

  Fim = 0;
  CyclePrepare();

  while ( !Fim ) {
    i = poll( Fds, 1, INFTIM );
    if ( i < 0 ) {
      if ( errno == EINTR )
        continue;
      perror( "poll" );
      exit( 1 );
    }
    else if ( i == 0 ) {
      printf( "Humm, poll retornou sem dado para ser lido...\n" );
      continue;
    }
    if ( Fds[0].revents != 0 ) {
      /*
      Fico em loop para receber o maximo de pacotes enquanto houver prontos
      para receber.  Talvez uma melhor opcao fosse usar o modulo bufmod do
```



```
      STREAMS do Solaris.  A versao nova deixou o modulo muito mais pratico
      para se usar.  Nao vou usar por que isso poderia tornar a medida irreal.
      */
      while ( 1 ) {
        Flags = 0;

        if ( ( RecvLen = recv( dd, MsgDataBuffer, sizeof MsgDataBuffer, 0 ) ) <
0 ) {
          if ( errno == EAGAIN ) {
            /* Nada pronto para ler, voltar ao loop de poll()           */
            break;
          }
          perror( "recv" );
          InternalError();
        }

        /***************************************************************\

        A cada pacote, tenho que checar a maquina de estados interna
        e externa.  Isso e’ meio complicado, mas acho que torna o
        programa mais generico.

        Possibilidades:

        - Pacote fora de ordem (perdi alguns pacotes)
        - Ultimo pacote pode ter sido perdido
        - Pacote de retaguarda (ignorar caso o ultimo tenha sido visto)
          - Posso tentar medir o tempo de retaguarda ?
            Talvez isso seja util para medir o tempo de buffer de saída.
        - Pacote normal (posso tentar um fast detection ?)

        \***************************************************************/

        if ( mp->Sequence == 1 ) {
          /* Inicio de um novo ciclo...                                 */
          if ( Count != 0 ) {
            /*  O ciclo anterior nao terminou !!!                       */
            /*  Nem mesmo o pacote de retaguarda chegou ???             */
            StopTiming();
            printf( "Aviso: Ciclo nao terminado !\n ");
            NPackets = Count;
            CycleEnd( Magic );
          }
          StartTiming();
          Magic = mp->Magic;
          Count = Next = mp->Sequence + 1;
          NPackets = mp->NPackets;
          MsgSize = mp->Size;
          if ( NPackets == 0 ) {
            printf( "Recebendo Magic 0x%08lX-%02d: Size %4d, Time %d\n",
              Magic, Instance, mp->Size, mp->Time );
          }
          else {
            printf( "Recebendo Magic 0x%08lX-%02d: Size %4d, Total %ld\n",
```

```
            Magic, Instance, mp->Size, NPackets );
        }
        if ( mp->RealSize != RecvLen ) {
          printf( "Warning, wrong size: %d (%d)\n", RecvLen, mp->RealSize );
        }
        continue;
      }

      /*  Nao é o inicio de um novo ciclo, então deve ser continuação.  */

      if ( Count == 0 ) {
        /* Pode ser um pacote de retaguarda que não foi necessário.   */
        if ( mp->Sequence == ( mp->NPackets + 1 ) ) {
          continue;
        }
        /* Opa...  Perdi alguma coisa...                              */
        printf( "Pacote recebido (0x%08lX:%ld), ciclo não inicializado...\n",
                mp->Magic,
                mp->Sequence );
        /*  O que fazer nessa situacao ???                            */
        /*  Será que eu perdi o pacote 0, por conta de alguma         */
        /*  Multiplexação ???                                         */
        continue;
      }

      /*  Será que é o fim do ciclo ?                                 */
      if ( ( mp->NPackets != 0 ) && ( mp->Sequence >= mp->NPackets ) ) {
        NPackets = mp->NPackets;
        StopTiming();
        if ( mp->Sequence == mp->NPackets ) {
          /*  Fim da sequência, sem perda do último pacote.           */
        } else if ( mp->Sequence == ( mp->NPackets + 1 ) ) {
          /*  Perdi o último pacote, mas peguei a retaguarda.         */
          Add2Lost( Next, mp->Sequence-1 );
          printf( "Ciclo 0x%08lX-%02X: Salvo na retaguarda.\n", Magic, mp->Ins
tance );
        } else {
          /*  Epa, isso não deveria acontecer !!!                     */
          printf( "Erro: Número de sequência inválido: 0x%08lX:%ld\n",
            mp->Magic, mp->Sequence );
          CyclePrepare();
          continue;
        }

        /* Fim de um ciclo...                                         */
        CycleEnd( Magic );
        continue;
      }
      /*  Default: Nao faz nada ?  */

      if ( mp->Sequence != Next ) {
        Add2Lost( Next, mp->Sequence );
      }
      /*  Como fazer um FastPath para esse caso ???                   */
```



```
        ++Count;
        Next = mp->Sequence + 1;
      }
    }
  }
}


int
main( int argc, const char **argv )
{
  int n, error;
  pid_t pid;


  /*  Parsing do comando de linha.                                      */
  /*  Isso tá uma eca, mas TCP/IP de fazer melhor.                      */
  if ( argc > 1 ) {
    SetMuxOption( argv[1] );
  }
  ProgramSetup();
  PreSync();

  printf( "Opção de dispositivo \"%s\"\n", "udp" );
  printf( "Opção de multiplexação \"%s\"\n", mux_option->option );
  printf( "Hostname: %s\n", hostname );
  printf( "Data file: %s\n", DataFilename );

  if ( OpenLog() != 0 ) exit( 1 );

  #if USE_PRIOCNTL
  if ( SetRtPrio( 5, 0.100 ) != 0 ) {
    perror( "SetRtPrio" );
    InternalError();
  }
  printf( "Operando em prioridade de tempo real\n" );
  #endif

  printf( "Inicializando os processos filhos: " );
  fflush( stdout );

  /*  Abre as multiplas instâncias para enviar com multiplexação.       */
  for ( n = 0 ; n < mux_option->nproc ; ++n ) {
    if ( ( pid = fork() ) == 0 ) {
      /*  Filho...   */

      /* Posso mudar o device_option no filho, pois já fiz o fork()     */
      Master = 0;
      Instance = n+1;
      Sap += n * SAP_SOMA;
      initdevice();
      signal( SIGINT,  SIG_IGN );
      signal( SIGUSR1, SigHandler );
      sleep( 1 );              /*  Sincroniza com os irmaos             */
```

```
      MainLoop();
      cleardevice();
      exit( 0 );
    }
    if ( pid == -1 ) {
      perror( "fork" );
      exit( 1 );
      /*  Vou deixar um bando de filho rodando, não posso fazer nada.   */
      /*  Talvez, tentar montar um session id e matar todo mundo ?      */
      /*  Nao gostei.  Mudando o session Id eu perco o controle via     */
      /*  terminal.                                                     */
    }
    Filhos[n] = pid;
    printf( "." );
    fflush( stdout );
  }
  printf( "\n" );

  /*  Espera o sinal para terminar o programa...  */
  signal( SIGINT, SigHandler );
  pause();

  for ( n = 0 ; n < mux_option->nproc ; ++n ) {
    kill( Filhos[n], SIGTERM );
  }

  /*  Espera por todos os filhos.                                       */
  /*  Não gostei, mas tá funcionando, tá bom.                           */
  for ( n = 0 ; n <= mux_option->nproc ; ++n ) {
    error = wait( &error );  /* XXX - Eca...  */
  }
  if ( error != -1 || errno != ECHILD ) {
    printf( "Opa...\n" );
  }

  CloseLog();

  return 0;
}
```



```
#!/bin/sh

#
#  Script para rodar no transmissor, com varias medidas seguidas
#
#  IPC: Nao devo usar o NFS pra nada, se nao pode incluir nas medidas
#

KillIt()
{
  if [ "$lastpid" != "" ] ; then
    kill -TERM -$lastpid
  fi
  exit
}


PreCopy()
{
  if [ ! -d "$destdir" ] ; then
    mkdir "$destdir"
  fi
  if [ ! -d "$destdir"/data ] ; then
    mkdir "$destdir"/data
  else
    echo "Opa, ja existe um diretorio de dados..."
  fi
  cp sender "$destdir"
  cp sender-udp "$destdir"
  cp script "$destdir"
}

Prepare()
{
  medidas="nomux                            \
        mux2   mux4   mux8   mux16      \
        mux4p  \
        mux4b  \
        mux4bp \
        "
        # mux2   mux4   mux8   mux16    \
        # mux2p  mux4p  mux8p  mux16p   \
        # mux2b  mux4b  mux8b  mux16b   \
        # mux2bp mux4bp mux8bp mux16bp  \
        # mux2r  mux4r  mux8r  mux16r   \
        # mux2rb mux4rb mux8rb mux16rb  \
        # mux2br mux4br mux8br mux16br  \
        # mux2rx mux4rx mux8rx mux16rx  \

  echo -n "Sleeping for 10 seconds"
  sleep 10
  echo ""
  getmem -v 20
  echo -n "Sleeping for 10 seconds"
```

```
  sleep 10
  echo ""
}

DoOnce()
{
  "$destdir"/sender -bf -r 4 -s 30 $senderflags -d "$destdir"/data "$@" &
  lastpid=$!
  wait
  lastpid=""
}

DoOnceUdp()
{
  "$destdir"/sender-udp -bf -r 4 -s 30 $senderflags -d "$destdir"/data "$@" &
  lastpid=$!
  wait
  lastpid=""
}

RunNetwork()
{
  echo "Usando as flags \"$senderflags\" no programa sender."
  for a in $medidas ; do
    date
    DoOnce le $a
    date
    DoOnce llc $a
    date
    DoOnce snap $a
    date
    DoOnceUdp buzios.snap $a
  done
  date
}

RunLocal()
{
  echo "Usando as flags \"$senderflags\" no programa sender."
  #  O loopback do Jonny nao tem suporte a multiplexacao.
  date
  DoOnce lb nomux
  for a in $medidas ; do
    date
    DoOnce lbllc $a
    date
    DoOnce lbsnap $a
    date
    DoOnceUdp 10.127.0.2 $a
  done
  date
}

Usage()
```



```
{
  echo "Usage: script le|lb [sender options]"
}

Main()
{
  trap KillIt 2 3 15

  ../mux/checkplumb || exit 1

  destdir=/var/jonny
  if [ "$0" != "$destdir"/script ] ; then
    PreCopy
    exec "$destdir"/script "$@"
  fi

  mode="$1"
  if [ "$mode" = "" ] ; then
    Usage
    exit 1
  fi
  shift
  if [ "$#" = 0 ] ; then
    senderflags="-s 2 -r 1"
  else
    senderflags="$@"
  fi

  case "$mode" in
    le)
      Prepare
      RunNetwork 2>&1 | tee "$destdir"/data/script.log
      ;;
    lb)
      Prepare
      RunLocal 2>&1 | tee "$destdir"/data/script.log
      ;;
    *)
      Usage
      exit
      ;;
  esac
}

Main "$@"
```