# Le HOWTO Linux de la programmation SCSI

Heiko Eißfeldt heiko@colossus.escape.de (version française : Thierry Danis thierry.danis@hol.fr, le 26 Janvier 1998, largement basée sur la traduction de la version v1.4 faite par Bernard Choppy choppy@imaginet.fr).
v1.5, 7 Juin 1996

Ce document traite de la programmation de l'interface SCSI générique de Linux.

## Table des matières

**1 Les nouveautés** **3**

**2 Introduction** **3**

**3 Qu'est-ce que l'interface SCSI générique ?** **3**

**4 Que faut-il pour l'utiliser ?** **4**

4.1 Configuration du noyau . . . . . 4

4.2 Fichiers spéciaux . . . . . 4

4.3 Organisation des périphériques . . . . . 5

4.3.1 Insertion et retrait dynamiques de périphériques SCSI . . . . . 5

**5 Le guide du programmeur** **6**

**6 Vue d'ensemble de la programmation des périphériques** **6**

**7 Ouverture du périphérique** **8**

**8 La structure d'en-tête** **8**

**9 Exemple de commande de requête** **11**

**10 Le « tampon SCSI »** **14**

**11 Exemple d'utilisation du tampon SCSI** **15**

**12 Fonctions ioctl** **17**

**13 Valeurs par défaut du pilote** **18**

13.1 Tailles de transfert . . . . . 18

13.2 Timeout et valeurs de réessais . . . . . 18

**14 Comment obtenir les spécifications SCSI?** **18**

**15 D'autres sources d'information** **19**

15.1 HOWTOs et FAQs . . . . . 19

15.2 La liste de messagerie . . . . . 19

15.3 Exemples de code . . . . . 19

**16 Autres choses utiles** **20**

16.1 Aides à l'écriture de pilotes de périphériques . . . . . 20

16.2 Utilitaires . . . . . 20

**17 Autres interfaces d'accès au SCSI** **21**

**18 Commentaires finals** **21**

**19 Remerciements** **21**

**A Annexe** **22**

**B Traitement d'erreurs** **22**

B.1 Décodage de l'état d'erreur . . . . . 22

B.2 Codes d'état . . . . . 23

B.3 Clefs du buffer SCSI . . . . . 25

B.4 Codes hôte . . . . . 27

B.5 Codes du pilote . . . . . 27

**C Codes et qualificateurs du buffer SCSI additionnels** **28**

C.1 ASC et ASCQ dans l'ordre numérique . . . . . 28

**D Référence rapide des commandes SCSI** **36**

**E Programmes d'exemple** **41**

# 1 Les nouveautés

Les interfaces des nouveaux noyaux ont un peu changé. Le chapitre 'rescrutation des périphériques' est concerné par ces changements. Il est maintenant possible d'ajouter et d'enlever des périphériques SCSI à chaud et à la volée.

A partir de la version 1.3.98, certains fichiers d'inclusion importants ont été déplacés ou découpés (sg.h and scsi.h).

Quelques bugs idiots ont été remplacés par d'autres.

# 2 Introduction

Ce document est un guide d'installation et de programmation de l'interface générique SCSI de Linux.

Il traite des prérequis du noyau, de l'organisation des périphériques, et de l'interaction de base avec ces derniers. Quelques exemples simples de programmation en C sont inclus. Pour de plus amples détails sur la norme SCSI et les informations associées, reportez-vous à l'annexe de ce document.

Note : la version texte simple de ce document ne dispose pas de référénces croisées (indiquées par “”).

# 3 Qu'est-ce que l'interface SCSI générique ?

L'interface générique SCSI a été faite pour fournir un accès général à des périphériques SCSI (éventuellement exotiques). Elle a été développée par Lawrence Foard (`entropy@world.std.com`) et sponsorisée par Killy Corporation (voir les commentaires du fichier `drivers/scsi/sg.h`).

Cette interface permet à des programmes applicatifs (c'est-à-dire hors du noyau) d'accéder aux fonctionnalités de certains périphériques. Le développement de pilotes dans le noyau, plus risqués et difficiles à mettre au point, n'est ainsi plus nécessaire.

Néanmoins, si le périphérique n'est pas correctement programmé, il est possible de bloquer le bus SCSI, le pilote, ou le noyau. C'est pourquoi il est important de programmer correctement le pilote générique, et de commencer par sauvegarder tous les fichiers afin d'éviter une perte de données. Une autre précaution utile est de faire un `sync` avant de lancer vos programmes pour garantir l'écriture de tous les tampons sur le disque ; cela limitera la perte de données en cas de blocage du système.

Un autre avantage du pilote générique est que, aussi longtemps que l'interface elle-même ne change pas, toutes les applications restent indépendantes des nouveaux développements du noyau. En comparaison, les pilotes de bas niveau du noyau doivent suivre les évolutions internes de celui-ci.

Typiquement, le pilote générique est utilisé pour communiquer avec les nouveaux équipements SCSI qui exigent l'écriture d'applications utilisateur spécifiques pour tirer avantage de leurs fonctionalités (par exemple les scanners, les imprimantes, le juke-boxes de CDROM). L'interface générique permet un développement rapide de ces applications.

# 4 Que faut-il pour l'utiliser?

## 4.1 Configuration du noyau

Il vous faut évidemment un adapteur SCSI reconnu. De plus, votre noyau doit être compilé avec le support du pilote générique, en plus de celui pour votre adapteur. La configuration du noyau Linux (par `make config` sous `/usr/src/linux`) doit ressembler à:

```
 ...
*
* SCSI support
*
SCSI support? (CONFIG_SCSI) [n] y
*
* SCSI support type (disk, tape, CDrom)
*
 ...
Scsi generic support (CONFIG_CHR_DEV_SG) [n] y
*
* SCSI low-level drivers
*
 ...
```

Si le noyau est modulable, vous pouvez tout aussi bien utiliser les modules.

## 4.2 Fichiers spéciaux

Le pilote générique utilise ses propres fichiers spéciaux. Ils peuvent être créés par le script `MAKEDEV`, que l'on trouve habituellement dans le répertoire `/dev`. La commande `MAKEDEV sg` crée les fichiers suivants:

```
crw-------   1 root     system    21,   0 Aug 20 20:09 /dev/sga
crw-------   1 root     system    21,   1 Aug 20 20:09 /dev/sgb
crw-------   1 root     system    21,   2 Aug 20 20:09 /dev/sgc
crw-------   1 root     system    21,   3 Aug 20 20:09 /dev/sgd
crw-------   1 root     system    21,   4 Aug 20 20:09 /dev/sge
crw-------   1 root     system    21,   5 Aug 20 20:09 /dev/sgf
crw-------   1 root     system    21,   6 Aug 20 20:09 /dev/sgg
crw-------   1 root     system    21,   7 Aug 20 20:09 /dev/sgh
                                     |    |
                        numeros majeur, mineur
```

Notez que ces fichiers sont des fichiers en mode caractère pour permettre les accès directs. Sur certains systèmes, ces fichiers peuvent s'appeler `/dev/{sg0,sg1,...}`, en fonction de votre installation. Vous devrez donc adapter les exemples qui suivant en conséquence.

## 4.3 Organisation des périphériques

Les fichiers spéciaux sont affectés dynamiquement aux ID/LUN (LUN : Unité logique) du bus SCSI. Les périphériques sont consécutivement alloués selon les unités logiques de chaque périphérique détecté lors de la scrutation du bus, les plus petits LUN/ID/BUS étant alloués en premier. Le noyau commence par le premier contrôleur SCSI et continue sans interruption avec tous les autres adapteurs. Cette étape est actuellement réalisée lors de l'initialisation du pilote SCSI.

Par exemple, si vous avez trois périphériques SCSI configurés avec les ID 1, 3 et 5 sur le premier bus SCSI (chacun avec une seule unité logique), l'affectation sera la suivante :

```
/dev/sga -> SCSI id 1
/dev/sgb -> SCSI id 3
/dev/sgc -> SCSI id 5
```

Si vous ajoutez maintenant un nouveau périphérique d'identificateur 4, l'organisation après la prochaine scrutation du bus sera :

```
/dev/sga -> SCSI id 1
/dev/sgb -> SCSI id 3
/dev/sgc -> SCSI id 4
/dev/sgd -> SCSI id 5
```

Notez le changement pour l'identificateur 5 - le périphérique correspondant n'est plus affecté à `/dev/sgc`, mais à `/dev/sgd`.

Les noyaux plus récents permettent de changer cet ordre.

### 4.3.1 Insertion et retrait dynamiques de périphériques SCSI

Dans les noyaux récents avec un système de fichier `/proc` monté, il est possible de retirer et d'ajouter un périphérique libre (non-busy) à la volée.

Pour enlever un périphérique SCSI :

```
echo "scsi remove-single-device a b c d" > /proc/scsi/scsi
```

De la même manière, ajouter un périphérique SCSI se fera par :

```
echo "scsi add-single-device a b c d" > /proc/scsi/scsi
```

Ici, a, b, c et d sont définis de la façon suivante :

```
a == identificateur de l'adapteur (le premier a l'id 0)
b == canal SCSI sur l'adapteur (le premier a le numero 0)
c == ID
d == LUN (la premiere ayant le numero 0)
```

Ainsi, si nous désirons intervertir l'affectation des fichiers `/dev/sgc` et `/dev/sgd` de l'exemple précédent, nous pouvons faire :

```
echo "scsi remove-single-device 0 0 4 0" > /proc/scsi/scsi
echo "scsi remove-single-device 0 0 5 0" > /proc/scsi/scsi
echo "scsi add-single-device 0 0 5 0" > /proc/scsi/scsi
echo "scsi add-single-device 0 0 4 0" > /proc/scsi/scsi
```

puisque les périphériques génériques sont alloués dans leur ordre d'insertion.

Si vous voulez ajouter de nouveaux périphériques sur le bus SCSI, gardez à l'esprit qu'un nombre limité d'entrées supplémentaires a été attribué. La mémoire a été allouée au démarrage, et il n'y a de place que pour 2 entrées supplémentaires.

# 5 Le guide du programmeur

Les sections qui suivent s'adressent aux programmeurs désireux d'utiliser l'interface générique SCSI dans leurs propres applications. Nous allons donner un exemple permettant d'accéder à un périphérique SCSI par le biais des commandes INQUIRY et TESTUNITREADY.

Lors de l'utilisation de ces exemples, prenez garde à ce qui suit :

- l'emplacement des fichiers d'inclusion `sg.h` et `scsi.h` a changé à partir du noyau 1.3.98. Ces fichiers se trouvent maintenant à `/usr/src/linux/include/scsi`, qui devrait être un lien vers `/usr/include/scsi`. Dans les versions précédentes, ils se trouvaient dans `/usr/src/linux/drivers/scsi`. Nous supposerons dans la suite que vous utilisez un de ces noyaux récents.
- l'interface générique SCSI a été étendue dans la version 1.1.68 du noyau. Les exemples nécessitent au moins cette version. En revanche, évitez d'utiliser les noyaux de 1.1.77 à 1.1.89 qui disposent d'une interface générique SCSI défectueuse.
- la constante DEVICE de la section qui décrit le périphérique accédé doit être positionnée en fonction de vos périphériques disponibles (reportez-vous au chapitre 8).

# 6 Vue d'ensemble de la programmation des périphériques

Le fichier d'inclusion `include/scsi/sg.h` contient une description de l'interface (celle du noyau 1.3.98) :

```
struct sg_header
 {
                        /*
                         * longueur du paquet entrant (y compris en-tete)
                         */
  int pack_len;
                        /*
```

```
                        * taille max de la reponse attendue
                        */
int reply_len;
                       /*
                        * numero d'id du paquet
                        */
int pack_id;
                       /*
                        * 0 == ok,
                        * pour les autres, voir les codes pour errno
                        */
int result;
                       /*
                        * Force la longueur a 12 pour les commandes des
                        * groupes 6 & 7
                        */
unsigned int twelve_byte:1;
                       /*
                        * pour utilisation future
                        */
unsigned int other_flags:31;
                       /*
                        * uniquement utilise lors des lectures
                        */
unsigned char sense_buffer[16];
                       /*
                        * la commande suit puis les donnees de la
                        * commande
                        * ..............
                        */
```

Cette structure décrit comment une commande SCSI doit être traitée et disposer de place pour le résultat de son exécution. Les composants individuels de la structure seront abordés plus loin à la section 8.

La méthode générale pour échanger des données avec le pilote générique est la suivante : pour envoyer une commande à un périphérique générique ouvert, il faut écrire (`write()`) un bloc composé des trois parties suivantes :

```
                    struct sg_header
                      commande SCSI
              donnees envoyees avec la commande
```

Pour obtenir le résultat d'une commande, il faut lire (`read()`) un bloc composés des parties suivantes (similaires à l'écriture) :

```
                    struct sg_header
                    donnees en entree
```

Il s'agit d'une vue générale de la procédure. Les sections qui suivent décrivent chaque étape en détail.

NOTE : jusqu'à de récentes versions du noyau, il était nécessaire de bloquer le signal SIGINT entre les appels `write()` et le `read()` correspondant (par exemple, par `sigprocmask()`). Un retour après la partie écriture sans lecture pour récupérer les résultats va bloquer les accès suivants. Le blocage du signal n'a pas encore été inclus dans le code des exemples. Evitez donc d'envoyer un SIGINT (par ^C, par exemple) lors du test de ceux-ci.

# 7 Ouverture du périphérique

Un périphérique générique doit être ouvert avant tout accès en lecture ou en écriture :

```
int fd = open(nom_du_peripherique, O_RDWR);
```

(ce qui précède s'applique aussi pour les matériels en lecture seule comme les lecteurs de CDROM).

Il faut exécuter un `write` pour envoyer la commande et un `read` pour en lire le résultat. En cas d'erreur, le code de retour est négatif (se reporter à la section B pour la liste complète des codes de retour).

# 8 La structure d'en-tête

La structure d'en-tête `struct sg_header` est utilisée comme couche de contrôle entre l'application et le pilote du noyau. Abordons maintenant le détail de ses composants.

**int pack_len**

définit la taille du bloc envoyé au pilote. Cette valeur est définie dans le noyau pour une utilisation interne.

**int reply_len**

définit la taille du bloc accepté en réponse. Cette valeur est définie du côté application.

**int pack_id**

Ce champ facilite l'appariement des réponses aux requêtes. L'application peut fournir un identifiant unique à chaque requête. Supposons que vous ayez écrit un certain nombre de commandes (disons 4) pour un périphérique. Celles-ci peuvent fonctionner en parallèle, l'une d'entre elles étant la plus rapide. Lors de la lecture des réponses par quatre « read », celles-ci ne sont pas forcément dans l'ordre des requêtes. Pour identifier la réponse correcte pour une requête, on peut utiliser le champ `pack_id`. Habituellement, cette valeur est incrémentée après chaque requête (et boucle éventuellement). Le nombre maximum de requêtes émises simultanément est limité par le noyau à SG_MAX_QUEUE (en général, quatre).

**int result**

C'est la valeur du résultat d'un appel à `read` ou à `write`. Elle est (parfois) définie par la le pilote générique (partie noyau). Il est plus prudent de le positionner à 9 avant l'appel à `write`. Ces codes sont déclarés dans le fichier `errno.h` (0 indique un résultat correct).

**unsigned int twelve_byte:1**

Ce champ n'est nécessaire que lors de l'utilisation de commandes spécifiques non standard (dans la plage 0xc0 à 0xff). Lorsque la longueur de ces commandes est de 12 octets au lieu de 10, il faut positionner ce champ à 1 avant l'appel à `write`. D'autres longueurs de commandes ne peuvent être utilisées. Ce champ est positionné par l'application.

**unsigned char sense_buffer[16 ]**

Ce tampon est positionné après l'exécution d'une commande (après un appel à `read()`) et contient le code de « sensation » SCSI (SCSI send code. NdT. : dans le reste du document, on utilisera simplement la formule « tampon SCSI »). Certains résultats de commandes doivent être lus à cet emplacement (par exemple pour `TESTUNITREADY`). Il ne contient habituellement que des octets nuls. La valeur de ce champ est positionnée par le pilote générique (partie noyau).

L'exemple de fonction qui suit s'interface directement avec le pilote générique du noyau. Il définit la structure d'en-tête, envoie la commande par `write`, lit le résultat par `read` et effectue un nombre (limité) de contrôles d'erreurs. Les données du tampon SCSI sont disponibles dans le tampon de sortie (sauf si un pointeur nul a été fourni, auquel cas elles se trouvent dans le tampon d'entrée). Nous l'utiliserons dans les exemples qui suivent.

Note: positionnez la valeur de `DEVICE` à celle qui correspond à votre matériel.

```
#define DEVICE "/dev/sgc"

/* Programme d'exemple utilisant l'interface SCSI generique */
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#include <errno.h>
#include <scsi/sg.h>


#define SCSI_OFF sizeof(struct sg_header)
static  unsigned char cmd[SCSI_OFF + 18];       /* tampon de commande SCSI */
int     fd;                                     /*
                                                 * descripteur de peripherique/
                                                 * fichier SCSI
                                                 */
/* traite une commande SCSI complete. Utilise l'interface generique */
static int handle_SCSI_cmd(unsigned cmd_len,     /* longueur de commande  */
                           unsigned in_size,     /* taille data en entree */
                           unsigned char *i_buff,/* tampon d'entree       *//
                           unsigned out_size,    /* taille data en sortie */
                           unsigned char *o_buff /* tampon de sortie      */
                           )
{
    int status = 0;
```

```
    struct sg_header *sg_hd;

    /* verifications de securite */
    if (!cmd_len) return -1;            /* necessite que cmd_len != 0 */
    if (!i_buff) return -1;             /* necessite que i_buff != NULL */

#ifdef SG_BIG_BUFF
    if (SCSI_OFF + cmd_len + in_size > SG_BIG_BUFF) return -1;
    if (SCSI_OFF + out_size > SG_BIG_BUFF) return -1;
#else
    if (SCSI_OFF + cmd_len + in_size > 4096) return -1;
    if (SCSI_OFF + out_size > 4096) return -1;
#endif

    if (!o_buff) out_size = 0;          /* pas de tampon de sortie, pas de */
                                        /* taille                          */

    /* construction de l'en-tete generique de peripherique */
    sg_hd = (struct sg_header *) i_buff;
    sg_hd->reply_len   = SCSI_OFF + out_size;
    sg_hd->twelve_byte = cmd_len == 12;
    sg_hd->result = 0;
#if     0
    sg_hd->pack_len    = SCSI_OFF + cmd_len + in_size; /* non indispensable */
    sg_hd->pack_id;     /* inutilise */
    sg_hd->other_flags; /* inutilise */
#endif

    /* envoi de la commande */
    status = write( fd, i_buff, SCSI_OFF + cmd_len + in_size );
    if ( status < 0 || status != SCSI_OFF + cmd_len + in_size ||
                       sg_hd->result ) {
        /* condition d'erreur */
        fprintf( stderr, "write(generic) resultat = 0x%x cmd = 0x%x\n",
                    sg_hd->result, i_buff[SCSI_OFF] );
        perror("");
        return status;
    }

    if (!o_buff) o_buff = i_buff; /* controle du pointeur du tampon */

    /* recuperation du resultat */
    status = read( fd, o_buff, SCSI_OFF + out_size);
    if ( status < 0 || status != SCSI_OFF + out_size || sg_hd->result ) {
        /* condition d'erreur */
        fprintf( stderr, "read(generic) statut = 0x%x, resultat = 0x%x, "
                         "cmd = 0x%x\n",
                         status, sg_hd->result, o_buff[SCSI_OFF] );
```

```
        fprintf( stderr, "read(generic) tampon SCSI "
                "%x %x %x %x %x %x %x %x %x %x %x %x %x %x %x %x\n",
                sg_hd->sense_buffer[0],         sg_hd->sense_buffer[1],
                sg_hd->sense_buffer[2],         sg_hd->sense_buffer[3],
                sg_hd->sense_buffer[4],         sg_hd->sense_buffer[5],
                sg_hd->sense_buffer[6],         sg_hd->sense_buffer[7],
                sg_hd->sense_buffer[8],         sg_hd->sense_buffer[9],
                sg_hd->sense_buffer[10],        sg_hd->sense_buffer[11],
                sg_hd->sense_buffer[12],        sg_hd->sense_buffer[13],
                sg_hd->sense_buffer[14],        sg_hd->sense_buffer[15]);
        if (status < 0)
            perror("");
    }
    /* A-t-on ce qu'on attendait ? */
    if (status == SCSI_OFF + out_size) status = 0; /* on a tout */

    return status;  /* 0 indique que tout est bon */
}
```

Bien que cela puisse sembler quelque peu complexe au premier abord, une grande partie du code est dédiée aux contrôle et détection d'erreurs (ce qui est utile même une fois que le code fonctionne correctement).

`Handle_SCSI_cmd` présente une forme généralisée pour tous les types de commandes SCSI, qui correspondent à l'une des catégories qui suivent :

| Mode de donnees | Exemple de commande |
|---|---|
| ni entree ni sortie de donnees | test d'unite prete |
| pas d'entree, sortie de donnees | requete, lecture |
| entree de donnees, pas de sortie | selection de mode, ecriture |
| entree et sortie de donnees | detection de mode |

# 9 Exemple de commande de requête

L'une des commandes SCSI de base est `INQUIRY`, utilisée pour identifier les type et constructeur du périphérique. Voici la définition issue de la spécification SCSI-2 (se reporter au standard SCSI-2 pour les détails).

Table 44: Commande INQUIRY

| Bit / Octet | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Code operation (12h) | | | | | | | |
| 1 | Numero d'unite logique | | | | Reserve | | | EVPD |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2 | Code page | | | | | | | |
| 3 | Reserve | | | | | | | |
| 4 | Taille d'allocation | | | | | | | |
| 5 | Controle | | | | | | | |

Les données en sortie ont l'allure suivante:

Table 45: Format standard de donnees INQUIRY

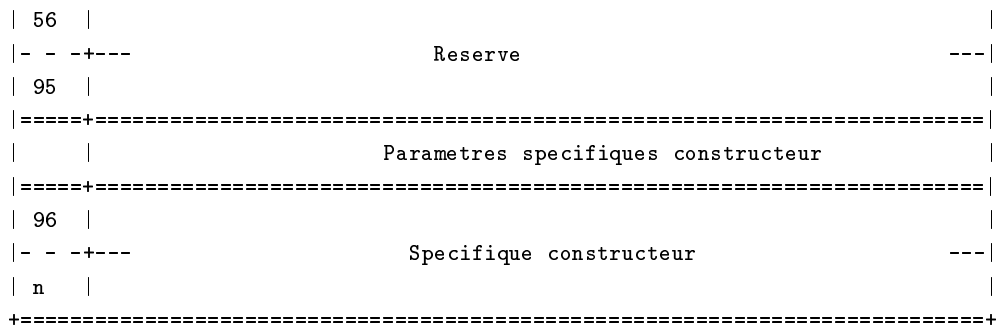
| Bit Octet | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Qualificateur de periph. | | | Type de peripherique | | | | |
| 1 | RMB | | | Modificateur de type de peripherique | | | | |
| 2 | Version ISO | | | Version ECMA | | | Version approuvee ANSI | |
| 3 | AENC | TrmIOP | | Reserve | | | Format de donnees en reponse | |
| 4 | Longueur additionnelle (n-4) | | | | | | | |
| 5 | Reserve | | | | | | | |
| 6 | Reserve | | | | | | | |
| 7 | RelAdr | WBus32 | WBus16 | Sync | Linked | Reserve | CmdQue | SftRe |
| 8 | (MSB) | | | | | | | |
| - - | --- | | | | | | | |
| 15 | | | | | | | | (LSB) |
| 16 | (MSB) | | | | | | | |
| - - | --- | | | | | | | |
| 31 | | | | | | | | (LSB) |
| 32 | (MSB) | | | | | | | |
| - - | --- | | | | | | | |
| 35 | | | | | | | | (LSB) |
| 36 | | | | | | | | |
| - - | --- | | | | | | | |
| 55 | | | | | | | | |

Identification de constructeur

Identification de produit

Niveau de revision du produit

Specifique constructeur




L'exemple qui suit utilise la fonction de bas niveau `handle_SCSI_cmd` pour effectuer la commande SCSI INQUIRY.

Tout d'abord, nous ajoutons le bloc de commande à l'en-tête générique, puis appelons `handle_SCSI_cmd`. Notez que l'argument taille de tampon en sortie de l'appel `handle_SCSI_cmd` exclut la taille de l'en-tête générique. Après l'exécution de la commande, le tampon de sortie contient les informations, sauf si une erreur s'est produite.

```
#define INQUIRY_CMD     0x12
#define INQUIRY_CMDLEN  6
#define INQUIRY_REPLY_LEN 96
#define INQUIRY_VENDOR  8       /* decalage vers le nom du constructeur */

/* recherche du constructeur et du modele */
static unsigned char *Inquiry ( void )
{
  unsigned char Inqbuffer[ SCSI_OFF + INQUIRY_REPLY_LEN ];
  unsigned char cmdblk [ INQUIRY_CMDLEN ] =
      { INQUIRY_CMD,  /* commande                */
                0,  /* lun/reserve             */
                0,  /* code de page            */
                0,  /* reserve                 */
  INQUIRY_REPLY_LEN,  /* longueur allocation     */
                0 };/* reserve / drapeau / lien */

  memcpy( cmd + SCSI_OFF, cmdblk, sizeof(cmdblk) );

  /*
   * +------------------+
   * | struct sg_header | <- commande
   * +------------------+
   * | copie de cmdblk  | <- commande + SCSI_OFF
   * +------------------+
   */

  if (handle_SCSI_cmd(sizeof(cmdblk), 0, cmd,
                      sizeof(Inqbuffer) - SCSI_OFF, Inqbuffer )) {
```

```
        fprintf( stderr, "La requete a echoue\n" );
        exit(2);
    }
    return (Inqbuffer + SCSI_OFF);
}
```

L'exemple ci-dessus suit cette structure. La fonction Inquiry recopie son bloc de commande après l'en-tête générique (donné par `SCSI_OFF`). Les données en entrée sont absentes de cette commande. `handle_SCSI_cmd` définit la structure d'en-tête. Nous pouvons maintenant implémenter la fonction `main` qui complète ce programme d'exemple fonctionnel.

```
void main( void )
{
  fd = open(DEVICE, O_RDWR);
  if (fd < 0) {
    fprintf( stderr, "Il faut les permissions lecture/ecriture pour "DEVICE".\n" );
    exit(1);
  }

  /* affiche certains champs du resultat de Inquiry() */
  printf( "%s\n", Inquiry() + INQUIRY_VENDOR );
}
```

Tout d'abord, nous ouvrons le périphérique, contrôlons l'absence d'erreur, puis appelons la fonction de haut niveau. Ensuite, nous affichons des résultats sous une forme lisible, dont le constructeur, le produit et la version.

Note : il y a plus d'informations dans le résultat de « Inquiry » que ce que fournit ce petit programme. Il vous est loisible d'étendre celui-ci au type de périphérique, version ANSI, etc. Le type de périphérique a une importance particulière, puisqu'il détermine les jeux de commandes obligatoires et facultatives pour celui-ci. Si vous ne souhaitez pas le programmer vous-même, Eric Youngdale a réalisé le programme scsiinfo, qui fournit à peu près toute information disponible pour un périphérique SCSI. Cherchez sur tsx-11.mit.edu dans `pub/Linux/ALPHA/scsi` (NdT : on trouvera ce programme sur les sites miroirs français, comme ftp.ibp.fr, à un emplacement similaire).

# 10 Le « tampon SCSI »

Les commandes qui ne renvoient pas de données peuvent fournir des informations d'état à l'aide du tampon SCSI (qui fait partie intégrante de la structure d'en-tête). Les données d'état sont disponibles lorsque la commande précédente s'est terminée avec un statut « CHECK CONDITION ». Dans ce cas, le noyau rapatrie automatiquement les données d'état à l'aide d'une commande « REQUEST SENSE ». Sa structure est la suivante :

```
+=====-========-========-========-========-========-========-========-========+
|  Bit|   7    |   6    |   5    |   4    |   3    |   2    |   1    |   0    |
```

| Octet | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | Valide | Code d'erreur (70h ou 71h) | | | | | | |
| 1 | | Numero de segment | | | | | | |
| 2 | Filemark | EOM | ILI | Reserve | Clef d'etat | | | |
| 3 | (MSB) | | | | | | | |
| - - | - | Information | | | | | | |
| 6 | | | | | | | | (LSB) |
| 7 | | Longueur additionnelle d'etat (n-7) | | | | | | |
| 8 | (MSB) | | | | | | | |
| - - | - | Information specifique de la commande | | | | | | |
| 11 | | | | | | | | (LSB) |
| 12 | | Code d'etat additionnel | | | | | | |
| 13 | | Qualificateur de code d'etat additionnel | | | | | | |
| 14 | | Code d'unite de champ remplacable | | | | | | |
| 15 | SKSV | | | | | | | |
| - - | - | Specifique clef d'etat | | | | | | |
| 17 | | | | | | | | |
| 18 | | | | | | | | |
| - - | - | Octets supplementaires d'etat | | | | | | |
| n | | | | | | | | |

Note : les champs les plus utiles sont la clef d'état (cf. section B.3), le code d'état additionnel et le qualificateur de code d'état additionnel (cf. section C). Les deux derniers sont utilisés en combinaison l'un avec l'autre.

# 11 Exemple d'utilisation du tampon SCSI

Nous allons utiliser ici la commande « TEST UNIT READY » pour contrôler si un support est chargé dans notre périphérique. Les déclarations d'en-tête et la fonction `handle_SCSI_cmd` de l'exemple de « Inquiry » seront aussi nécessaires.

Table 73: Commande TEST UNIT READY

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Octet | | | | | | | | |

| | | |
|---|---|---|
| 0 | | Code operation (00h) |
| 1 | Numero d'unite logique LUN | Reserve |
| 2 | | Reserve |
| 3 | | Reserve |
| 4 | | Reserve |
| 5 | | Controle |

Voici la fonction qui l'implémente:

```
#define TESTUNITREADY_CMD 0
#define TESTUNITREADY_CMDLEN 6

#define ADD_SENSECODE 12
#define ADD_SC_QUALIFIER 13
#define NO_MEDIA_SC 0x3a
#define NO_MEDIA_SCQ 0x00

int TestForMedium ( void )
{
  /* demande le statut READY */
  static unsigned char cmdblk [TESTUNITREADY_CMDLEN] = {
      TESTUNITREADY_CMD, /* commande */
                      0, /* lun/reserve */
                      0, /* reserve */
                      0, /* reserve */
                      0, /* reserve */
                      0};/* controle */

  memcpy( cmd + SCSI_OFF, cmdblk, sizeof(cmdblk) );

  /*
   * +------------------+
   * | struct sg_header | <- commande
   * +------------------+
   * | copie de cmdblk  | <- commande + SCSI_OFF
   * +------------------+
   */

  if (handle_SCSI_cmd(sizeof(cmdblk), 0, cmd,
                           0, NULL)) {
      fprintf (stderr, "Unite non prete\n");
```

```
        exit(2);
    }

    return
     *(((struct sg_header*)cmd)->sense_buffer +ADD_SENSECODE) !=
                                                    NO_MEDIA_SC ||
     *(((struct sg_header*)cmd)->sense_buffer +ADD_SC_QUALIFIER) !=
                                                    NO_MEDIA_SCQ;
}
```

Nous pouvons maintenant réaliser le contrôle à l'aide de la fonction `main`:

```
void main( void )
{
  fd = open(DEVICE, O_RDWR);
  if (fd < 0) {
    fprintf( stderr, "Il faut les permissions lecture/ecriture pour "DEVICE".\n"
 );
    exit(1);
  }

  /* on regarde si le support est charge */

  if (!TestForMedium()) {
    printf("le support n'est pas charge\n");
  } else {
    printf("le support est charge\n");
  }
}
```

Le fichier `generic_demo.c` en annexe contient les deux exemples.

# 12 Fonctions ioctl

Deux fonctions ioctl sont disponibles :

- `ioctl(fd, SG_SET_TIMEOUT, &Timeout)`; définit la valeur du timeout à `Timeout` * 10 millisecondes. `Timeout` doit être déclaré en tant qu'entier (int).
- `ioctl(fd, SG_GET_TIMEOUT, &Timeout)`; lit la valeur du timeout en cours. `Timeout` doit être déclaré en tant qu'entier (int).

# 13 Valeurs par défaut du pilote

## 13.1 Tailles de transfert

Actuellement (au moins jusqu'au noyau version 1.1.68), les tailles d'entrée et de sortie doivent être inférieures ou égales à 4096 octets, sauf si le noyau a été compilé avec la constante `SG_BIG_BUFF` définie, auquel cas elles sont limitées à `SG_BIG_BUFF` (soit 32768) octets. Les tailles données comprennent l'en-tête générique ainsi que le bloc de commande fourni en entrée. `SG_BIG_BUFF` peut être augmentée sans problème jusqu'à (131072 - 512). Pour en bénéficier, vous devrez bien évidemment regénérer un nouveau noyau et redémarrer avec.

## 13.2 Timeout et valeurs de réessais

La valeur du timeout par défaut est d'une minute (`Timeout` = 6 000). Elle peut être modifiée à l'aide d'un appel à ioctl (cf. section 12). Le nombre de réessais par défaut est un.

# 14 Comment obtenir les spécifications SCSI ?

Il existe des normes appelées SCSI-1, SCSI-2 et SCSI-3. Les normes assurent à peu de choses près la compatibilité ascendante.

Le standard SCSI-1 est (d'après l'auteur) en grande partie obsolète, et SCSI-2 est celui qui est le plus largement utilisé. SCSI-3 est très jeune et très cher. Ces jeux de commandes normalisés définissent des commandes obligatoires et facultatives pour les constructeurs de matériels SCSI et doivent être préférées aux extensions spécifiques non normalisées et pour lesquelles l'information est plus difficile à obtenir. Evidemment, il n'y a parfois aucune alternative à ces extensions propriétaires.

Des copies électroniques sont disponibles par FTP anonyme depuis :

- ftp.cs.tulane.edu:pub/scsi
- ftp.symbios.com:/pub/standards
- ftp.cs.uni-sb.de:/pub/misc/doc/scsi

(J'ai eu mes spécifications SCSI dans le CD-ROM Linux d'Yggdrasil, dans le répertoires /usr/doc/scsi-2 et /usr/doc/scsi-1).

La FAQ SCSI liste aussi les sources d'information imprimée suivantes :

Les spécifications SCSI - disponible depuis :

```
Global Engineering Documents
15 Inverness Way East
Englewood Co  80112-5704
(800) 854-7179
  SCSI-1: X3.131-1986
  SCSI-2: X3.131-199x
```

```
        SCSI-3 X3T9.2/91-010R4 Working Draft

(Global Engineering Documentation in Irvine, CA (714)261-1455??)

SCSI-1: Doc \# X3.131-1986 from ANSI, 1430 Broadway, NY, NY 10018

IN-DEPTH EXPLORATION OF SCSI peut etre trouve chez
Solution Technology, Attn: SCSI Publications, POB 104, Boulder Creek,
CA 95006, (408)338-4285, FAX (408)338-4374

THE SCSI ENCYLOPEDIA et SCSI BENCH REFERENCE peuvent etre obtenus chez
ENDL Publishing, 14426 Black Walnut Ct., Saratoga, CA 95090,
(408)867-6642, FAX (408)867-2115

SCSI: UNDERSTANDING THE SMALL COMPUTER SYSTEM INTERFACE est publie chez
Prentice-Hall, ISBN 0-13-796855-8
```

# 15 D'autres sources d'information

## 15.1 HOWTOs et FAQs

Le **SCSI-HOWTO** Linux de Drew Eckhardt (NdT : disponible en version française) traite de tous les contrôleurs SCSI reconnus ainsi que des questions spécifiques aux périphériques. De nombreux aides pour le dépannage sont fournies. Il est disponible sur sunsite.unc.edu dans /pub/Linux/docs/LDP et sur ses sites miroirs.

Les questions générales concernant le SCSI ont une réponse dans la FAQ SCSI du groupe de news comp.periphs.scsi (disponible sur tsx-11 dans `/pub/linux/ALPHA/scsi` et sur les sites miroirs).

## 15.2 La liste de messagerie

Il existe une **liste de messagerie** qui traite des rapports d'anomalies et questions sur le développement SCSI sous Linux. Pour la rejoindre, envoyez un courrier à `majordomo@vger.rutgers.edu` avec la ligne `subscribe linux-scsi` dans le corps du message. Les messages doivent être envoyés à `linux-scsi@vger.rutgers.edu`. Un texte d'aide peut être demandé par envoi de la ligne de message « help » à `majordomo@vger.rutgers.edu`.

## 15.3 Exemples de code

**sunsite.unc.edu: apps/graphics/hpscanpbm-0.3a.tar.gz**

Ce paquetage gère un scanner HP scanjet à l'aide de l'interface générique.

**tsx-11.mit.edu: BETA/cdrom/private/mkisofs/cdwrite-1.3.tar.gz**

Le paquetage cdwrite utilise l'interface générique pour écrire une image de CD sur un graveur.

**sunsite.unc.edu: apps/sound/cds/cdda2wav*.src.tar.gz**

Un composant pour mes propres applications, qui copie des pistes audio de CD sous forme de fichiers wav.

# 16 Autres choses utiles

Des choses qui pourraient devenir pratiques. Je n'ai aucune idée de la présence de versions plus récentes ou meilleures ici ou là. Toute information est la bienvenue.

## 16.1 Aides à l'écriture de pilotes de périphériques

Ces documents peuvent être trouvés sur le serveur ftp de sunsite.unc.edu et sur ses miroirs.

**/pub/Linux/docs/kernel/kernel-hackers-guide**

Le guide des stakhanovistes du noyau LDP (NdT : Projet de Documentation Linux). Il est peut-être un peu ancien, mais traite les points les plus fondamentaux.

**/pub/Linux/docs/kernel/drivers.doc.z**

Ce document traite de l'écriture de pilotes caractères.

**/pub/Linux/docs/kernel/tutorial.doc.z**

Tutoriel sur l'écriture d'un pilote de périphérique caractère avec le code.

**/pub/Linux/docs/kernel/scsi.paper.tar.gz**

Un document Latex décrivant comment écrire un pilote SCSI.

**/pub/Linux/docs/hardware/DEVICES**

Une liste des numéros majeurs et mineurs utilisés par Linux.

## 16.2 Utilitaires

**tsx-11.mit.edu: ALPHA/scsi/scsiinfo*.tar.gz**

Programme d'interrogation d'un périphérique SCSI pour obtenir ses paramètres d'utilisation, listes de défauts, etc. Une interface X nécessitant Tk/Tcl/wish est disponible. Avec cette dernière, vous pouvez facilement modifier la configuration du lecteur.

**tsx-11.mit.edu: ALPHA/kdebug**

Une extension à pour le déverminage du noyau.

# 17 Autres interfaces d'accès au SCSI

Sous Linux, on peut accéder différemment au SCSI via des appels ioctl SCSI_IOCTL_SEND_COMMAND qui nécessitent des privilèges root. Les paquetages « scsiinfos » ainsi que « cdda2wav » les utilisent.

D'autres interfaces similaires sont utilisées dans le monde Unix, mais ne sont pas disponibles pour Linux :

1. CAM (Common Access Method) développée par Future Domain et d'autres constructeurs SCSI. Linux dispose maintenant d'un petit support pour un système CAM SCSI (essentiellement pour bouter depuis un disque dur). CAM supporte même le mode « target », qui permet de déguiser un ordinateur en périphérique SCSI (c.à.d. réaliser un petit réseau SCSI).
2. ASPI (Advanced SCSI Programming Interface) developpée par Adaptec. C'est le standard de facto pour les machines MS-DOS.
3. ??? est disponible sur NeXTStep.
4. DSLIB est disponible sur Silicon Graphics.
5. SCSI... est disponible sur les machines SUN.
6. SCO Unix a aussi quelque chose.
7. HPUX utilise des ioctl. –¿

D'autres interfaces applicatives existent aussi chez SCO(TM), NeXT(TM), Silicon Graphics(TM) et SUN(TM).

# 18 Commentaires finals

L'interface générique SCSI jette un pont sur le fossé entre les applications utilisateur et les périphériques spécifiques. Mais plutôt que de charger de nombreux programmes avec des jeux similaires de fonctions de bas niveau, il serait plus souhaitable de disposer d'une bibliothèque partagée avec un jeu généralisé de fonctions de bas niveau pour un usage particulier. Le but principal est de disposer de couches d'interfaces indépendantes. Une bonne conception doit séparer une application en routines de bas niveau et indépendantes du matériel. Celles-ci peuvent être placées dans une bibliothèque partagée et rendues disponibles pour toutes les applications. Ainsi, les interfaces standardisées doivent être suivies autant que possible avant d'en réaliser de nouvelles.

Vous devriez maintenant en savoir plus que moi sur l'interface générique SCSI de Linux. Vous pouvez donc commencer à développer de puissantes applications pour le plus grand bénéfice de la communauté Linux...

# 19 Remerciements

Un grand merci à Jeff Tranter pour ses corrections et améliorations considérables de ce texte, ainsi qu'à Carlos Puchol pour ses commentaires utiles. L'aide de Drew Eckhardt et Eric Youngdale sur mes premières questions (idiotes) sur l'utilisation de cette interface a été appréciée.

# A Annexe

# B Traitement d'erreurs

Les fonctions `open`, `ioctl`, `write` et `read` peuvent renvoyer des erreurs. Dans ce cas, leur valeur de retour est -1 et la variable globale errno est positionnée au numéro d'erreur (négatif). Les valeurs de errno sont définies dans `/usr/include/errno.h`. Les valeurs négatives possibles sont les suivantes :

| Fonction | Erreur | Description |
|---|---|---|
| open | ENXIO | peripherique invalide |
| | EACCES | l'acces n'est pas en lecture/ecriture (O_RDWR) |
| | EBUSY | le peripherique est accede en mode non bloquant, mais il est occupe actuellement |
| | ERESTARTSYS | erreur interne. Essayez de la rendre reproductible et informez-en le canal SCSI (pour les details sur le rapport de bogue, se reporter au SCSI-HOWTO de Drew Eckhardts). |
| ioctl | ENXIO | peripherique invalide |
| read | EAGAIN | le peripherique bloque. Essayez plus tard. |
| | ERESTARTSYS | erreur interne. Essayez de la rendre reproductible et informez-en le canal SCSI (pour les details sur le rapport de bogue, se reporter au SCSI-HOWTO de Drew Eckhardts). |
| write | EIO | taille trop petite (plus petite que cette de l'en-tete generique). Attention : il n'y a actuellement aucun controle de debordement. |
| | EAGAIN | le peripherique bloque. Essayez plus tard. |
| | ENOMEM | la memoire necessaire pour cette requete ne peut etre allouee. Essayez plus tard sauf si vous depassez la taille maximale de transfert (cf. ci-dessus). |
| select | | sans description |
| close | | sans description |

Pour la lecture et l'écriture, des valeurs de retour positivent indiquent comme d'habitude la quantité d'octets transférés. Cette valeur doit correspondre à celle demandée.

## B.1 Décodage de l'état d'erreur

En plus, une information détaillée est fournie par `hd_status` du noyau et par `sense_buffer` du périphérique (cf. section ref id="sec-sensebuff" name="Le tampon SCSI"¿), les deux utilisant la structure d'en-tête générique.

Les différents sens de `hd_status` peuvent être trouvés dans `drivers/scsi/scsi.h`. Cet `unsigned int` est composé de différentes parties :

| lsb | ... | ... | msb |
|---|---|---|---|

```
=======|===========|===========|============
status | sense key | host code | driver byte
```

Les macros de `drivers/scsi/scsi.h` sont disponibles, mais elles ne peuvent malheureusement pas être facilement utilisées à cause d'interdépendances tordues entre fichiers d'en-tête. Il faudrait faire une passe sur ces fichiers pour clarifier les choses.

```
        Macro          | Description
=======================|============================================
status_byte(hd_status) | Etat du peripherique. cf. section Codes d'etat
msg_byte(hd_status)    | du peripherique. cf. section buffer SCSI
host_byte(hd_status)   | du noyau. cf. section codes hote
driver_byte(hd_status) | du noyau. cf. section codes intermediaires
```

## B.2 Codes d'état

Les codes d'état de périphérique qui suivent (issus de `drivers/scsi/scsi.h`) sont disponibles :

```
Valeur | Symbole
=======|====================
0x00   | GOOD
0x01   | CHECK_CONDITION
0x02   | CONDITION_GOOD
0x04   | BUSY
0x08   | INTERMEDIATE_GOOD
0x0a   | INTERMEDIATE_C_GOOD
0x0c   | RESERVATION_CONFLICT
```

On constate que ces valeurs symboliques ont subi un **décalage droit**. Lorsque l'état indique CHECK_CONDITION, les données du buffer SCSI sont valides (contrôlez en particulier le code d'état additionnel et le qualificateur de code d'état additionnel).

Les valeurs qui suivent concernent les spécifications SCSI-2 :

Table 27 : Code de l'octet d'etat

| Bits de l'octet d'etat | | | | | | | | Etat |
|---|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| R | R | 0 | 0 | 0 | 0 | 0 | R | GOOD |
| R | R | 0 | 0 | 0 | 0 | 1 | R | CHECK CONDITION |
| R | R | 0 | 0 | 0 | 1 | 0 | R | CONDITION MET |
| R | R | 0 | 0 | 1 | 0 | 0 | R | BUSY |
| R | R | 0 | 1 | 0 | 0 | 0 | R | INTERMEDIATE |
| R | R | 0 | 1 | 0 | 1 | 0 | R | INTERMEDIATE-CONDITION MET |
| R | R | 0 | 1 | 1 | 0 | 0 | R | RESERVATION CONFLICT |

| R | R | 1 | 0 | 0 | 0 | 1 | R | COMMAND TERMINATED |
|---|---|---|---|---|---|---|---|---|
| R | R | 1 | 0 | 1 | 0 | 0 | R | QUEUE FULL |
| Tous autres codes | | | | | | | | Reserve |

R = Bit reserve

La définition des codes de l'octet d'état sont données ci-dessous :

**GOOD.**

Cet état indique que la cible a correctement exécuté la commande.

**CHECK CONDITION.**

Cet état indique qu'une condition de contention s'est produite (cf. 6.6).

**CONDITION MET.**

Cet état, ou INTERMEDIATE-CONDITION MET est renvoyé lorsque les conditions de l'opération demandée sont satisfaites (cf. commandes SEARCH DATA et PRE-FETCH).

**BUSY.**

Cet état indique que la cible est occupée. Il peut être renvoyé lorsque la cible ne peut accepter de commande depuis un initiateur inacceptable par ailleurs (i.e. conflit d'absence de réservation). L'action de reprise recommandée est une nouvelle tentative ultérieure.

**INTERMEDIATE.**

Cet état, ou INTERMEDIATE-CONDITION MET doit être renvoyée après chaque commande réussie d'une série de commandes liées (sauf pour la dernière), sauf si celle-ci se termine par un CHECK CONDITION, RESERVATION CONFLICT, ou COMMAND TERMINATED. Si ni INTERMEDIATE ni INTERMEDIATE-CONDITION MET n'est renvoyé, la série de commandes se termine, ainsi que le processus d'entrées/sorties.

**INTERMEDIATE-CONDITION MET.**

Cet état est la combinaison de CONDITION MET et de INTERMEDIATE.

**RESERVATION CONFLICT.**

Cet état doit être renvoyé lorsqu'un initiateur tente d'accéder à une unité logique ou à un extension à l'intérieur d'une unité logique réservée avec un type de réservation en conflit pour un autre périphérique SCSI (cf. commandes RESERVE et RESERVE UNIT). L'action de reprise recommandée est une nouvelle tentative ultérieure.

**COMMAND TERMINATED.**

Cet état doit être renvoyé lorsque la cible termine le processus d'entrées/sorties après réception d'un message TERMINATE I/O PROCESS (cf. 5.6.22). Cet état indique aussi qu'une condition de contention s'est produite (cf. 6.6).

**QUEUE FULL.**

Cet état doit être implémenté si la file d'attente marquée (tagged queuing) l'est aussi. Il est renvoyé lors de la réception d'un message SIMPLE QUEUE TAG, ORDERED QUEUE TAG, ou HEAD OF QUEUE TAG et que la file de commandes est pleine. Le processus d'entrée/sortie n'est alors pas placé dans la file de commandes.

## B.3 Clefs du buffer SCSI

Les clefs résultantes peuvent être rapatriées à l'aide de la macro `msg_byte` (cf. section B.1). Les symboles du noyau qui suivent sont prédéfinis dans `drivers/scsi/scsi.h`:

```
Valeur | Symbole
=======|================
0x00   | NO_SENSE
0x01   | RECOVERED_ERROR
0x02   | NOT_READY
0x03   | MEDIUM_ERROR
0x04   | HARDWARE_ERROR
0x05   | ILLEGAL_REQUEST
0x06   | UNIT_ATTENTION
0x07   | DATA_PROTECT
0x08   | BLANK_CHECK
0x0a   | COPY_ABORTED
0x0b   | ABORTED_COMMAND
0x0d   | VOLUME_OVERFLOW
0x0e   | MISCOMPARE
```

Une liste extraite de la doc SCSI-2 suit (issue de la section 7.2.14.3) :

Table 69: Description des clefs (0h-7h) du buffer SCSI

| Clef | Description |
|---|---|
| 0h | NO SENSE. Indique qu'aucune information specifique n'est disponible pour l'unite logique designee. C'est le cas pour les commandes reussies ou celles dont l'etat est CHECK CONDITION ou COMMAND TERMINATED a cause de l'un des bits filemark, EOM ou ILI. |
| 1h | RECOVERED ERROR. Indique que la reussite de la derniere commande fut conditionnee par une action de reparation effectuee par la cible. Les octets additionnels peuvent fournir des details, ainsi que le champ information. Lorsque plusieurs erreurs reparees se produisent durant une commande, le choix de celle indiquee (premiere, derniere, plus severe, etc.) depend du peripherique. |
| 2h | NOT READY. Indique que l'unite logique est inaccessible. Une |

| | |
|---|---|
| | intervention manuelle peut etre necessaire. |
| 3h | MEDIUM ERROR. Indique la fin d'une commande sur une erreur non-recuperable, causee probablement par un defaut du support ou une erreur de donnees. Cette clef peut aussi etre renvoyee si la cible ne peut faire la distinction entre un defaut du support et un defaut specifique du materiel (clef 4h). |
| 4h | HARDWARE ERROR. Indique que la cible a detecte une erreur materielle irrecuperable (defaut du controleur, du peripherique, erreur de parite, etc.) lors de l'execution de la commande ou d'un auto-test. |
| 5h | ILLEGAL REQUEST. Indique qu'un parametre illegal a ete detecte dans le bloc de description de commande ou dans les parametres additionnels (pour certaines commandes : FORMAT UNIT, SEARCH DATA, etc.). Si la cible detecte un parametre incorrect, il doit terminer celle-ci sans modifier le contenu du support. Si le parametre incorrect se trouve dans les parametres additionnels, la cible peut avoir deja modifie le support. Cette clef est aussi renvoyee lors de la reception d'un message IDENTIFY invalide (5.6.7). |
| 6h | UNIT ATTENTION. Indique que le support amovible a pu etre change ou que la cible a ete reinitialisee. Cf. 6.9 pour d'autres information sur cette condition. |
| 7h | DATA PROTECT. Indique qu'une commande de lecture ou d'ecriture a ete tentee sur un bloc protege contre cette operation. Celle-ci n'est pas effectuee. |

Table 70: Description des clefs (8h-Fh) du buffer SCSI

| Clef | Description |
|---|---|
| 8h | BLANK CHECK. Indique qu'un peripherique a ecriture unique ou sequentiel a trouve un support vierge ou une indication de fin de donnees de formatage lors de la lecture, ou qu'un support non vierge a ecriture seule a ete trouve pendant l'ecriture. |
| 9h | Vendor Specific. Cette clef est disponible pour indiquer des cas particuliers specifiques du constructeur. |
| Ah | COPY ABORTED. Indique qu'une commande COPY, COMPARE ou COPY AND VERIFY a echoue a cause d'une condition d'erreur sur le peripherique source, destination ou les deux (cf. 7.2.3.2 pour plus de details). |

| | |
|---|---|
| Bh | ABORTED COMMAND. Indique que la cible a abandonne la commande. L'initiateur peut eventuellement corriger le probleme par une nouvelle tentative. |
| Ch | EQUAL. Indique qu'une commande SEARCH DATA a satisfait une condition d'egalite. |
| Dh | VOLUME OVERFLOW. Indique qu'un peripherique a memoire-tampon a atteint la fin de partition et que des donnees non ecrites sur le support peuvent rester dans le tampon. Une (ou plusieurs) commande RECOVER BUFFER DATA peut etre tentee pour lire les donnees non ecrites depuis le tampon. |
| Eh | MISCOMPARE. Indique que les donnees source ne correspondent pas a celles lues sur le support. |
| Fh | RESERVE. |

## B.4 Codes hôte

Les codes hôtes qui suivent sont définis au niveau de `drivers/scsi/scsi.h`. Ils sont positionnés par le pilote du noyau et doivent être utilisés avec la macro `host_byte` (cf. section B.1) :

| Valeur | Symbole | Description |
|---|---|---|
| 0x00 | DID_OK | Pas d'erreur |
| 0x01 | DID_NO_CONNECT | Connexion impossible avant le timeout |
| 0x02 | DID_BUS_BUSY | BUS occupe durant la periode de timeout |
| 0x03 | DID_TIME_OUT | Timeout atteint pour une autre raison |
| 0x04 | DID_BAD_TARGET | Mauvaise cible |
| 0x05 | DID_ABORT | Arret effectue pour une autre raison |
| 0x06 | DID_PARITY | Erreur de parite |
| 0x07 | DID_ERROR | Erreur interne |
| 0x08 | DID_RESET | Reinitialise par quelqu'un |
| 0x09 | DID_BAD_INTR | Interruption inattendue recue |

## B.5 Codes du pilote

Le pilote de niveau intermédiaire catégorise l'état renvoyé par le pilote de bas niveau en fonction du buffer SCSI du périphérique. Il suggère certaines actions pouvant être tentées comme un réessai, un abandon ou un changement de topographie. La routine `scsi_done` de `scsi.c` effectue un travail très différencié fondé sur `host_byte()`, `status_byte()`, `msg_byte()` et la suggestion précédente. Ensuite, il positionne l'octet du pilote afin d'indiquer ce qui a été réalisé. L'octet du pilote est en deux parties: l'état du pilote et la suggestion.

Chaque moitié est composée des valeurs suivantes (de `scsi.h`) combinées par un `OR`:

| Valeur | Symbole | Description ou etat du pilote |
|---|---|---|
| 0x00 | DRIVER_OK | pas d'erreur |
| 0x01 | DRIVER_BUSY | inutilise |
| 0x02 | DRIVER_SOFT | inutilise |
| 0x03 | DRIVER_MEDIA | inutilise |
| 0x04 | DRIVER_ERROR | erreur interne du pilote |
| 0x05 | DRIVER_INVALID | termine (DID_BAD_TARGET ou DID_ABORT) |
| 0x06 | DRIVER_TIMEOUT | termine avec timeout |
| 0x07 | DRIVER_HARD | termine avec une erreur fatale |
| 0x08 | DRIVER_SENSE | buffer SCSI disponible pour informations |

| Valeur | Symbole | Description de la suggestion |
|---|---|---|
| 0x10 | SUGGEST_RETRY | reessayer la requete SCSI |
| 0x20 | SUGGEST_ABORT | abandonner la requete |
| 0x30 | SUGGEST_REMAP | remape le bloc (non encore implemente) |
| 0x40 | SUGGEST_DIE | laisser le noyau tomber en "panic" |
| 0x80 | SUGGEST_SENSE | lire le buffer SCSI du peripherique |
| 0xff | SUGGEST_IS_OK | rien a faire |

# C Codes et qualificateurs du buffer SCSI additionnels

Lorsque l'état de la commande SCSI exécutée est CHECK_CONDITION, des données sont disponibles dans le buffer SCSI. Les code et qualificateur additionnels se trouvent dans ce tampon.

Je joins ici deux tables issues des spécifications SCSI-2. La première est triée alphabétiquement, la seconde, numériquement (NdT : la traduction ayant un tantinet bouleversé l'ordre alphabétique, seule la table triée par numéros a été conservée. Le lecteur pourra se reporter à la version originale en américain pour la liste alphabétique).

## C.1 ASC et ASCQ dans l'ordre numérique

La table qui suit fournit une liste de descriptions avec les périphériques auxquels elles s'appliquent.

Table 364 : Assignements ASC et ASCQ

| |
|---|
| D - peripherique a acces Direct (Disque) |
| .T - peripherique a acces sequenTiel (bande magneTique) |
| . I - Imprimante |
| . P - Processeur |
| . .W -WORM (CD-ROM inscriptible une fois) |
| . . R - CD-ROM (lecture seule) |

| | | |

```
|           .  . R - CD-ROM (lecture seule)                                  |
|           .  .  S - Scanner ou numeriseur                                  |
|           .  .  .O - memoire Optique                                       |
|           .  .  . M - changeur de Media                                    |
|           .  .  .  C - peripherique de Communications                      |
|           .  .  .  .                                                       |
| ASC ASCQ  DTIPWRSOMC  DESCRIPTION                                          |
| --- ----              ------------------------------------------------     |
|  0A  00   DTIPWRSOMC  debordement de la trace d'erreur                     |
|  0B  00                                                                    |
|  0C  00    T     S    erreur d'ecriture                                    |
|  0C  01   D   W  O    erreud d'ecriture corrigee par auto-reallocation     |
|  0C  02   D   W  O    erreur d'ecriture - auto-reallocation impossible     |
|  0D  00                                                                    |
|  0E  00                                                                    |
|  0F  00                                                                    |
|  10  00   D   W  O    erreur ID, CRC ou ECC                                |
|  11  00   DT  WRSO    erreur de lecture irrecuperable                      |
|  11  01   DT  W SO    nombre d'essais atteint                              |
|  11  02   DT  W SO    erreur trop longue a corriger                        |
|  11  03   DT  W SO    erreurs de lecture multiples                         |
|  11  04   D   W  O    erreur de lecture - auto-reallocation impossible     |
|  11  05       WR O    erreur irrecuperable L-EC                            |
|  11  06       WR O    erreur irrecuperable CIRC                            |
|  11  07       W  O    erreur de resynchronisation de donnees               |
|  11  08    T          lecture de bloc incomplete                           |
|  11  09    T          pas de breche trouvee                                |
|  11  0A   DT     O    erreur mal corrigee                                  |
|  11  0B   D   W  O    erreur de lecture - reassignement recommande         |
|  11  0C   D   W  O    erreur de lecture - reecriture recommandee           |
|  12  00   D   W  O    marque d'adresse introuvable pour le champ ID        |
|  13  00   D   W  O    marque d'adresse introuvable pour le champ donnees   |
|  14  00   DTI WRSO    identite enregistree introuvable                     |
|  14  01   DT  WR O    enregistrement introuvable                           |
|  14  02    T          marque de fichier ou de jeu introuvable              |
|  14  03    T          fin de donnees introuvable                           |
|  14  04    T          erreur de sequence de bloc                           |
|  15  00   DTI WRSOM   erreur de positionnement aleatoire                   |
|  15  01   DTI WRSOM   erreur de positionnement mecanique                   |
|  15  02   DT  WR O    erreur de positionnement detectee par la lecture     |
|  16  00   DW     O    erreur de marque de synchronisation de donnees       |
|  17  00   DT  WRSO    donnees recuperees sans correction d'erreur          |
|  17  01   DT  WRSO    donnees recuperees apres plusieurs essais            |
|  17  02   DT  WR O    donnees recuperees avec un decalage de tete positif  |
|  17  03   DT  WR O    donnees recuperees avec un decalage de tete negatif  |
|  17  04       WR O    donnees recuperees avec plusieurs essais et/ou CIRC  |
|  17  05   D   WR O    donnees recuperees sur l'ID de secteur precedent     |
|  17  06   D   W  O    donnees recuperees sans ECC - donnees auto-reallouees |
```

| | | | | | |
|---|---|---|---|---|---|
| 17 | 07 | D | W | O | donnees recuperees sans ECC - reassignement recommande |
| 17 | 08 | D | W | O | donnees recuperees sans ECC - reecriture recommandee |
| 18 | 00 | DT | WR | O | donnees recuperees avec correction d'erreur |
| 18 | 01 | D | WR | O | donnees recuperees avec correction & plusieurs essais |
| 18 | 02 | D | WR | O | donnees recuperees - donnees auto-reallouees |
| 18 | 03 | | R | | donnees recuperees avec CIRC |
| 18 | 04 | | R | | donnees recuperees avec LEC |
| 18 | 05 | D | WR | O | donnees recuperees - reassignement recommande |
| 18 | 06 | D | WR | O | donnees recuperees - reecriture recommandee |

Table 364 : (suite)

D - peripherique a acces Direct (Disque)
.T - peripherique a acces sequenTiel (bande magneTique)
. I - Imprimante
. P - Processeur
. .W -WORM (CD-ROM inscriptible une fois)
. . R - CD-ROM (lecture seule)
. . S - Scanner ou numeriseur
. . .O - memoire Optique
. . . M - changeur de Media
. . . C - peripherique de Communications
. . . .

| ASC | ASCQ | DTIPWRSOMC | DESCRIPTION |
|---|---|---|---|
| 19 | 00 | D O | erreur de liste de defauts |
| 19 | 01 | D O | liste de defauts indisponible |
| 19 | 02 | D O | erreur de liste de defauts en liste primaire |
| 19 | 03 | D O | erreur de liste de defauts en liste secondaire (grown) |
| 1A | 00 | DTIPWRSOMC | erreur de taille de la liste de defauts |
| 1B | 00 | DTIPWRSOMC | erreur de transfert de donnees synchrone |
| 1C | 00 | D O | liste de defauts introuvable |
| 1C | 01 | D O | liste de defauts primaire introuvable |
| 1C | 02 | D O | liste de defauts secondaire (grown) introuvable |
| 1D | 00 | D W O | erreur de comparaison durant la verification |
| 1E | 00 | D W O | ID recupere avec ECC |
| 1F | 00 | | |
| 20 | 00 | DTIPWRSOMC | code d'operation de commande incorrect |
| 21 | 00 | DT WR OM | adresse du bloc logique hors limites |
| 21 | 01 | M | adresse d'element incorrecte |
| 22 | 00 | D | fonction illegale (seulement 20 00, 24 00 ou 26 00) |
| 23 | 00 | | |
| 24 | 00 | DTIPWRSOMC | champ incorrect en CDB |
| 25 | 00 | DTIPWRSOMC | unite logique non supportee |
| 26 | 00 | DTIPWRSOMC | champ incorrect en liste de parametres |
| 26 | 01 | DTIPWRSOMC | parametre non supporte |
| 26 | 02 | DTIPWRSOMC | valeur de parametre incorrecte |

| | | | |
|---|---|---|---|
| 26 | 03 | DTIPWRSOMC | parametres de seuil non supportes |
| 27 | 00 | DT W O | protection en ecriture |
| 28 | 00 | DTIPWRSOMC | transition non-pret/pret (changement de medium ?) |
| 28 | 01 | M | acces a un element import ou export |
| 29 | 00 | DTIPWRSOMC | allumage, reinit. ou reinit. du bus a eu lieu |
| 2A | 00 | DTI WRSOMC | parametres changes |
| 2A | 01 | DTI WRSOMC | parametres de mode changes |
| 2A | 02 | DTI WRSOMC | parametres de trace changes |
| 2B | 00 | DTIPWRSO C | copie impossible : deconnexion du host impossible |
| 2C | 00 | DTIPWRSOMC | erreur de sequence de commandes |
| 2C | 01 | S | trop de fenetres specifiees |
| 2C | 02 | S | combinaison de fenetres incorrecte specifiee |
| 2D | 00 | T | erreur d'ecriture en ecrasement de donnees |
| 2E | 00 | | |
| 2F | 00 | DTIPWRSOMC | commandes annulees par un autre initiateur |
| 30 | 00 | DT WR OM | medium incompatible present |
| 30 | 01 | DT WR O | medium illisible - format inconnu |
| 30 | 02 | DT WR O | medium illisible - format incompatible |
| 30 | 03 | DT | cartouche de nettoyage presente |
| 31 | 00 | DT W O | format du medium endommage |
| 31 | 01 | D I O | echec de la commande de format |
| 32 | 00 | D W O | plus d'emplacement de defaut disponible |
| 32 | 01 | D W O | echec de mise a jour de la liste de defauts |
| 33 | 00 | T | erreur de longueur de bande |
| 34 | 00 | | |
| 35 | 00 | | |
| 36 | 00 | I | manque d'encre, de ruban ou de toner |

Table 364 : (suite)

D - peripherique a acces Direct (Disque)
.T - peripherique a acces sequenTiel (bande magneTique)
. I - Imprimante
. P - Processeur
. .W -WORM (CD-ROM inscriptible une fois)
. . R - CD-ROM (lecture seule)
. . S - Scanner ou numeriseur
. . .O - memoire Optique
. . . M - changeur de Media
. . . . C - peripherique de Communications
. . . .

| ASC | ASCQ | DTIPWRSOMC | DESCRIPTION |
|---|---|---|---|
| 37 | 00 | DTI WRSOMC | parametre arrondi |
| 38 | 00 | | |
| 39 | 00 | DTI WRSOMC | sauvegarde de parametres non supportee |
| 3A | 00 | DTI WRSOM | pas de medium |

| | | | | | |
|---|---|---|---|---|---|
| 3B | 00 | TI | | | erreur de positionnement sequentiel |
| 3B | 01 | T | | | erreur de positionnement de la bande au debut |
| 3B | 02 | T | | | erreur de positionnement de la bande a la fin |
| 3B | 03 | I | | | bande ou feuille-a-feuille non pret |
| 3B | 04 | I | | | erreur de SLEW (NdT : !?) |
| 3B | 05 | I | | | bourrage papier |
| 3B | 06 | I | | | haut de page non detecte |
| 3B | 07 | I | | | bas de page non detecte |
| 3B | 08 | T | | | erreur de repositionnement |
| 3B | 09 | | S | | lecture apres la fin du medium |
| 3B | 0A | | S | | lecture avant le debut du medium |
| 3B | 0B | | S | | position apres la fin du medium |
| 3B | 0C | | S | | position avant le debut du medium |
| 3B | 0D | | | M | emplacement de destination occupe |
| 3B | 0E | | | M | emplacement d'origine vide |
| 3C | 00 | | | | |
| 3D | 00 | DTIPWRSOMC | | | bits incorrects dans le message d'identification |
| 3E | 00 | DTIPWRSOMC | | | auto-configuration de l'unite non encore realisee |
| 3F | 00 | DTIPWRSOMC | | | les conditions de fonctionnement ont change |
| 3F | 01 | DTIPWRSOMC | | | le micro-code a ete change |
| 3F | 02 | DTIPWRSOMC | | | definition de fonctionnement modifiee |
| 3F | 03 | DTIPWRSOMC | | | les donnees de requete ont change |
| 40 | 00 | D | | | defaillance RAM (40nn obligatoire) |
| 40 | NN | DTIPWRSOMC | | | echec de diagnostic du composant nn (80h-FFh) |
| 41 | 00 | D | | | echec du chemin de donnees (40nn obligatoire) |
| 42 | 00 | D | | | echec d'alllumage ou d'auto-test (40nn obligatoire) |
| 43 | 00 | DTIPWRSOMC | | | erreur de message |
| 44 | 00 | DTIPWRSOMC | | | defaillance de cible interne |
| 45 | 00 | DTIPWRSOMC | | | echec de selection ou de reselection |
| 46 | 00 | DTIPWRSOMC | | | echec de la reinitialisation logicielle |
| 47 | 00 | DTIPWRSOMC | | | erreur de parite SCSI |
| 48 | 00 | DTIPWRSOMC | | | reception de message d'erreur detecte par initiateur |
| 49 | 00 | DTIPWRSOMC | | | erreur message incorrect |
| 4A | 00 | DTIPWRSOMC | | | erreur de phase de commande |
| 4B | 00 | DTIPWRSOMC | | | erreur de phase de donnees |
| 4C | 00 | DTIPWRSOMC | | | echec de l'auto-configuration de l'unite logique |
| 4D | 00 | | | | |
| 4E | 00 | DTIPWRSOMC | | | commandes en recouvrement |
| 4F | 00 | | | | |
| 50 | 00 | T | | | erreur d'ecriture en ajout |
| 50 | 01 | T | | | erreur de positionnement en ajout |
| 50 | 02 | T | | | erreur de positionnement par rapport au timing |
| 51 | 00 | T | O | | erreur d'effacement |
| 52 | 00 | T | | | defaut de cartouche |

Table 364 : (suite)

```
|           D - peripherique a acces Direct (Disque)                          |
|           .T - peripherique a acces sequenTiel (bande magneTique)           |
|           . I - Imprimante                                                  |
|           .  P - Processeur                                                 |
|           .  .W -WORM (CD-ROM inscriptible une fois)                        |
|           .  . R - CD-ROM (lecture seule)                                   |
|           .  .  S - Scanner ou numeriseur                                   |
|           .  .  .O - memoire Optique                                        |
|           .  .  . M - changeur de Media                                     |
|           .  .  .  C - peripherique de Communications                       |
|           .  .  .  .                                                       |
| ASC ASCQ  DTIPWRSOMC  DESCRIPTION                                           |
| --- ----              ------------------------------------------------      |
|  53  00   DTI WRSOM   echec de chargement ou d'ejection du medium           |
|  53  01    T          echec de dechargement de la bande                     |
|  53  02   DT  WR OM   peripherique protege contre le changement de medium   |
|  54  00      P        defaillance de l'interface host-SCSI                  |
|  55  00      P        defaut de ressources systeme                          |
|  56  00                                                                     |
|  57  00        R      impossible de recuperer la table du contenu           |
|  58  00          O    la generation n'existe pas                            |
|  59  00          O    lecture de bloc mis a jour                            |
|  5A  00   DTIPWRSOM   requete operateur ou demande de changement d'etat     |
|  5A  01   DT  WR OM   requete operateur d'extraction du medium              |
|  5A  02   DT  W  O    l'operateur a selectionne la protection en ecriture   |
|  5A  03   DT  W  O    l'operateur a selectionne l'autorisation d'ecriture   |
|  5B  00   DTIPWRSOM   exception de trace                                    |
|  5B  01   DTIPWRSOM   condition de seuil remplie                            |
|  5B  02   DTIPWRSOM   compteur de trace au maximum                          |
|  5B  03   DTIPWRSOM   plus de code pour la liste de trace                   |
|  5C  00   D       O   changement d'etat RPL                                 |
|  5C  01   D       O   SPINDLES synchronisees                                |
|  5C  02   D       O   SPINDLES non synchronisees                            |
|  5D  00                                                                     |
|  5E  00                                                                     |
|  5F  00                                                                     |
|  60  00         S     defaillance de la lampe                               |
|  61  00         S     erreur d'acquisition video                            |
|  61  01         S     impossible de capturer la video                       |
|  61  02         S     hors de la zone focalisee                             |
|  62  00         S     erreur de positionnement de la tete de digitalisation |
|  63  00        R      fin de zone utilisateur sur cette piste               |
|  64  00        R      mode illegal pour cette piste                         |
|  65  00                                                                     |
|  66  00                                                                     |
|  67  00                                                                     |
|  68  00                                                                     |
|  69  00                                                                     |
```

| | | |
|---|---|---|
| 6A | 00 | |
| 6B | 00 | |
| 6C | 00 | |
| 6D | 00 | |
| 6E | 00 | |
| 6F | 00 | |

Table 364 : (fin)

D - peripherique a acces Direct (Disque)
.T - peripherique a acces sequenTiel (bande magneTique)
. I - Imprimante
. P - Processeur
. .W -WORM (CD-ROM inscriptible une fois)
. . R - CD-ROM (lecture seule)
. . S - Scanner ou numeriseur
. . .O - memoire Optique
. . . M - changeur de Media
. . . . C - peripherique de Communications
. . . . .

| ASC | ASCQ | DTIPWRSOMC | DESCRIPTION |
|---|---|---|---|
| 70 | 00 | | |
| 71 | 00 | | |
| 72 | 00 | | |
| 73 | 00 | | |
| 74 | 00 | | |
| 75 | 00 | | |
| 76 | 00 | | |
| 77 | 00 | | |
| 78 | 00 | | |
| 79 | 00 | | |
| 7A | 00 | | |
| 7B | 00 | | |
| 7C | 00 | | |
| 7D | 00 | | |
| 7E | 00 | | |
| 7F | 00 | | |
| | | | |
| 80 | xxh \ | | |
| jusqu'a > | | | specifique constructeur |
| FF | xxh / | | |
| | | | |
| xxh | 80 \ | | |
| jusqu'a > | | | qualification du standard ASC specifique constructeur |
| xxh | FF / | | |
| | | | TOUS LES CODES VIDES OU NON MONTRES SONT RESERVES |

# D Référence rapide des commandes SCSI

La table 365 est une liste ordonnée numériquement des codes opération des commandes.

Table 365 : Codes operations SCSI-2

```
+=============================================================================+
|           D - peripherique a acces Direct               Clef de colonne     |
|           .T - peripherique a acces sequenTiel          N = Necessaire      |
|           . I - Imprimante                              O = Optionnel       |
|           .  P - Processeur                             C = Constructeur    |
|           .  .W -WORM (CD-ROM inscriptible une fois)    R = Reserve         |
|           .  . R - CD-ROM (lecture seule)                                   |
|           .  .  S - Scanner ou numeriseur                                   |
|           .  .  .O - memoire Optique                                        |
|           .  .  . M - changeur de Media                                     |
|           .  .  .  C - peripherique de Communications                       |
|           .  .  .  .                                                        |
|        OP DTIPWRSOMC Description                                            |
|----------+----------+-------------------------------------------------------|
|        00 NNNNNNNNNN test d'unite prete                                     |
|        01  N         rembobinage                                            |
|        01 O C OO OO  remise a zero de l'unite                               |
|        02 CCCCCC  C                                                         |
|        03 NNNNNNNNNN requete de buffer SCSI                                 |
|        04   O        formatage                                              |
|        04 N      O   formatage de l'unite                                   |
|        05 CNCCCC  C  lecture des limites de bloc                            |
|        06 CCCCCC  C                                                         |
|        07         O  initialisation de l'etat d'un element                  |
|        07 OCC O  OC  reassignation de blocs                                 |
|        08          N lecture de message (06)                                |
|        08 ONC OO OC  lecture (06)                                           |
|        08    O       reception                                              |
|        09 CCCCCC  C                                                         |
|        0A   N        impression                                             |
|        0A          N emission de message (06)                               |
|        0A    N       emission (06)                                          |
|        0A ON  O  OC  ecriture (06)                                          |
|        0B O   OO OC  deplacement (06)                                       |
|        0B   O        SLEW et impression                                     |
|        0C CCCCCC  C                                                         |
|        0D CCCCCC  C                                                         |
|        0E CCCCCC  C                                                         |
|        0F COCCCC  C  lecture inversee                                       |
```

| | | |
|---|---|---|
| 10 | O O | synchronisation du tampon |
| 10 | CN CCC | ecriture de marques de fichiers |
| 11 | CNCCCC | espace |
| 12 | NNNNNNNNNN | requete |
| 13 | COCCCC | verification (06) |
| 14 | COOCCC | recuperation des donnees bufferisees |
| 15 | ONO OOOOOO | selection de mode (06) |
| 16 | N NN NO | reservation |
| 16 | NN N | reservation d'unite |
| 17 | N NN NO | liberation |
| 17 | NN N | liberation d'unite |
| 18 | OOOOOOOO | copie |
| 19 | CNCCCC | effacement |
| 1A | ONO OOOOOO | lecture du buffer SCSI (06) |
| 1B | O | chargement dechargement |
| 1B | O | digitalisation |
| 1B | O | arret d'impression |
| 1B | O OO O | arret demarrage de l'unite |

Table 365 : (suite)

| | | |
|---|---|---|
| D - peripherique a acces Direct | | Clef de colonne |
| .T - peripherique a acces sequenTiel | | N = Necessaire |
| . I - Imprimante | | O = Optionnel |
| . P - Processeur | | C = Constructeur |
| . .W -WORM (CD-ROM inscriptible une fois) | | R = Reserve |
| . . R - CD-ROM (lecture seule) | | |
| . . S - Scanner ou numeriseur | | |
| . . .O - memoire Optique | | |
| . . . M - changeur de Media | | |
| . . . C - peripherique de Communications | | |
| . . . . | | |
| OP | DTIPWRSOMC | Description |
| 1C | OOOOOOOOOO | reception des resultats du diagnostic |
| 1D | NNNNNNNNNN | emission de diagnostic |
| 1E | OO OO OO | protection contre l'enlevement du medium |
| 1F | | |
| 20 | C CC C | |
| 21 | C CC C | |
| 22 | C CC C | |
| 23 | C CC C | |
| 24 | C CCN | definition de fenetre |
| 25 | O | lecture de fenetre |
| 25 | N N N | lecture de capacite |
| 25 | N | lecture de capacite de cd-rom |
| 26 | C CC | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 27 | C | | CC | | | | | lecture de message (10) |
| | 28 | | | | | | | O | lecture de message (10) |
| | 28 | N | | NNNN | | | | | lecture (10) |
| | 29 | C | | CC | O | | | | lecture de generation |
| | 2A | | | | | | | O | emission de message (10) |
| | 2A | | | | | | O | | emission (10) |
| | 2A | N | | N | | N | | | ecriture (10) |
| | 2B | | O | | | | | | localisation |
| | 2B | | | | | | O | | positionnement sur element |
| | 2B | O | | OO | O | | | | deplacement (10) |
| | 2C | C | | | | O | | | effacement (10) |
| | 2D | C | | O | | O | | | lecture de bloc mis a jour |
| | 2E | O | | O | | O | | | lecture et verification (10) |
| | 2F | O | | OO | O | | | | verification (10) |
| | 30 | O | | OO | O | | | | lecture de donnee haute (10) |
| | 31 | | | | | O | | | position d'objet |
| | 31 | O | | OO | O | | | | recherche de donnee egale (10) |
| | 32 | O | | OO | O | | | | recherche de donnee basse (10) |
| | 33 | O | | OO | O | | | | definition de limites (10) |
| | 34 | | | | | O | | | lecture de l'etat du tampon de donnees |
| | 34 | O | | OO | O | | | | pre-lecture |
| | 34 | | O | | | | | | lecture de position |
| | 35 | O | | OO | O | | | | synchronisation du cache |
| | 36 | O | | OO | O | | | | verrouillage/deverrouillage du cache |
| | 37 | O | | | | O | | | lecture de donnees defectueuses (10) |
| | 38 | | | | O | O | | | digitalisation du medium |
| | 39 | OOOOOOOO | | | | | | | comparaison |
| | 3A | OOOOOOOO | | | | | | | copie et verification |
| | 3B | OOOOOOOOOO | | | | | | | ecriture de tampon |
| | 3C | OOOOOOOOOO | | | | | | | lecture de tampon |
| | 3D | | | | O | O | | | mise a jour de bloc |
| | 3E | O | | OO | O | | | | lecture longue |
| | 3F | O | | O | O | | | | ecriture longue |

Table 365 : (suite)

| | | |
|---|---|---|
| D - peripherique a acces Direct | Clef de colonne | |
| .T - peripherique a acces sequenTiel | N = Necessaire | |
| . I - Imprimante | O = Optionnel | |
| . P - Processeur | C = Constructeur | |
| . .W -WORM (CD-ROM inscriptible une fois) | R = Reserve | |
| . . R - CD-ROM (lecture seule) | | |
| . . S - Scanner ou numeriseur | | |
| . . .O - memoire Optique | | |
| . . . M - changeur de Media | | |
| . . . C - peripherique de Communications | | |
| . . . . | | |

| OP | DTIPWRSOMC | Description |
|---|---|---|
| 40 | OOOOOOOOOO | changement de definition |
| 41 | O | ecriture identique |
| 42 | O | lecture de sous-canal |
| 43 | O | lecture du TOC |
| 44 | O | lecture d'en-tete |
| 45 | O | lecture audio (10) |
| 46 | | |
| 47 | O | lecture audio MSF |
| 48 | O | lecture d'index de piste audio |
| 49 | O | lecture de piste relative (10) |
| 4A | | |
| 4B | O | reprise de pause |
| 4C | OOOOOOOOOO | trace de selection |
| 4D | OOOOOOOOOO | trace du buffer SCSI |
| 4E | | |
| 4F | | |
| 50 | | |
| 51 | | |
| 52 | | |
| 53 | | |
| 54 | | |
| 55 | OOO OOOOOO | mode de selection (10) |
| 56 | | |
| 57 | | |
| 58 | | |
| 59 | | |
| 5A | OOO OOOOOO | mode du buffer SCSI (10) |
| 5B | | |
| 5C | | |
| 5D | | |
| 5E | | |
| 5F | | |

Table 365 : (fin)

| | |
|---|---|
| D - peripherique a acces Direct | Clef de colonne |
| .T - peripherique a acces sequenTiel | N = Necessaire |
| . I - Imprimante | O = Optionnel |
| . P - Processeur | C = Constructeur |
| . .W -WORM (CD-ROM inscriptible une fois) | R = Reserve |
| . . R - CD-ROM (lecture seule) | |
| . . S - Scanner ou numeriseur | |
| . . .O - memoire Optique | |
| . . . M - changeur de Media | |
| . . . C - peripherique de Communications | |

```
|           .  .  .  .                                                        |
|        OP DTLPWRSOMC Description                                             |
|----------+----------+-----------------------------------------------------|
|        A0                                                                    |
|        A1                                                                    |
|        A2                                                                    |
|        A3                                                                    |
|        A4                                                                    |
|        A5         N  deplacement de medium                                   |
|        A5      O     lecture audio (12)                                      |
|        A6         O  changement de medium                                    |
|        A7                                                                    |
|        A8          O lecture de message (12)                                 |
|        A8     OO O   lecture (12)                                            |
|        A9      O     lecture de piste relative (12)                          |
|        AA          O emission de message (12)                                |
|        AA     O  O   ecriture (12)                                           |
|        AB                                                                    |
|        AC        O   effacement (12)                                         |
|        AD                                                                    |
|        AE     O  O   ecriture et verification (12)                           |
|        AF     OO O   verification (12)                                       |
|        B0     OO O   recherche de donnee haute (12)                          |
|        B1     OO O   recherche de donnee egale (12)                          |
|        B2     OO O   recherche de donnee basse (12)                          |
|        B3     OO O   definition des limites (12)                             |
|        B4                                                                    |
|        B5                                                                    |
|        B5         O  demande d'adresse d'element volume                      |
|        B6                                                                    |
|        B6         O  emission de TAG de volume                               |
|        B7        O   lecture des donnees de defauts (12)                     |
|        B8                                                                    |
|        B8         O  lecture de l'etat d'element                             |
|        B9                                                                    |
|        BA                                                                    |
|        BB                                                                    |
|        BC                                                                    |
|        BD                                                                    |
|        BE                                                                    |
|        BF                                                                    |
+==============================================================================+
```

# E Programmes d'exemple

Voici le programme exemple en C qui demande le constructeur et le modèle et indique si un medium est chargé dans le périphérique.

```
#define DEVICE "/dev/sgc"
/* Programme de demonstration de l'interface SCSI generique */
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#include <errno.h>
#include <scsi/sg.h>

#define SCSI_OFF sizeof(struct sg_header)
static unsigned char cmd[SCSI_OFF + 18];        /* tampon de commandes SCSI */
int     fd;                           /* descripteur de periph./fichier SCSI */

/* traite une commande SCSI complete. Utilise l'interface SCSI generique */

static int handle_scsi_cmd(unsigned cmd_len,         /* longueur */
                           unsigned in_size,         /* taille data IN */
                           unsigned char *i_buff,    /* tampon IN */
                           unsigned out_size,        /* taille data OUT */
                           unsigned char *o_buff     /* tampon OUT */
                           )
{
    int status = 0;
    struct sg_header *sg_hd;

    /* quelques controles de routine */
    if (!cmd_len) return -1;            /* cmd_len doit etre != 0 */
    if (!i_buff) return -1;             /* tampon IN doit etre != NULL */
#ifdef SG_BIG_BUFF
    if (SCSI_OFF + cmd_len + in_size > SG_BIG_BUFF) return -1;
    if (SCSI_OFF + out_size > SG_BIG_BUFF) return -1;
#else
    if (SCSI_OFF + cmd_len + in_size > 4096) return -1;
    if (SCSI_OFF + out_size > 4096) return -1;
#endif

    if (!o_buff) out_size = 0;

    /* construction de l'en-tete du pilote generique */
    sg_hd = (struct sg_header *) i_buff;
    sg_hd->reply_len   = SCSI_OFF + out_size;
    sg_hd->twelve_byte = cmd_len == 12;
    sg_hd->result = 0;
```

```
#if     0
    sg_hd->pack_len    = SCSI_OFF + cmd_len + in_size; /* pas indispensable */
    sg_hd->pack_id;     /* inutilise */
    sg_hd->other_flags; /* inutilise */
#endif

    /* envoi de la commande */
    status = write( fd, i_buff, SCSI_OFF + cmd_len + in_size );
    if ( status < 0 || status != SCSI_OFF + cmd_len + in_size ||
                       sg_hd->result ) {
        /* une erreur s'est produite */
        fprintf( stderr, "ecriture (generique) resultat = 0x%x cmd = 0x%x\n",
                    sg_hd->result, i_buff[SCSI_OFF] );
        perror("");
        return status;
    }

    if (!o_buff) o_buff = i_buff;       /* controle du pointeur du tampon */

    /* recuperation du resultat */
    status = read( fd, o_buff, SCSI_OFF + out_size);
    if ( status < 0 || status != SCSI_OFF + out_size || sg_hd->result ) {
        /* une erreur s'est produite */
        fprintf( stderr, "lecture (generique) resultat = 0x%x cmd = 0x%x\n",
                sg_hd->result, o_buff[SCSI_OFF] );
        fprintf( stderr, "read(generic) sense "
                "%x %x %x %x %x %x %x %x %x %x %x %x %x %x %x %x\n",
                sg_hd->sense_buffer[0],         sg_hd->sense_buffer[1],
                sg_hd->sense_buffer[2],         sg_hd->sense_buffer[3],
                sg_hd->sense_buffer[4],         sg_hd->sense_buffer[5],
                sg_hd->sense_buffer[6],         sg_hd->sense_buffer[7],
                sg_hd->sense_buffer[8],         sg_hd->sense_buffer[9],
                sg_hd->sense_buffer[10],        sg_hd->sense_buffer[11],
                sg_hd->sense_buffer[12],        sg_hd->sense_buffer[13],
                sg_hd->sense_buffer[14],        sg_hd->sense_buffer[15]);
        if (status < 0)
            perror("");
    }
    /* Voyons si nous avons ce que nous attendions */
    if (status == SCSI_OFF + out_size) status = 0; /* on a tout */

    return status;  /* 0 indique que tout est OK */
}

#define INQUIRY_CMD     0x12
#define INQUIRY_CMDLEN  6
#define INQUIRY_REPLY_LEN 96
#define INQUIRY_VENDOR  8    /* Decalage sur le constructeur dans la reponse */
```

```
/* On demande le constructeur et le modele */
static unsigned char *Inquiry ( void )
{
  unsigned char Inqbuffer[ SCSI_OFF + INQUIRY_REPLY_LEN ];
  unsigned char cmdblk [ INQUIRY_CMDLEN ] =
      { INQUIRY_CMD,  /* commande */
                  0,  /* lun/reserve */
                  0,  /* code page */
                  0,  /* reserve */
  INQUIRY_REPLY_LEN,  /* longueur d'allocation */
                  0 };/* reserve/drapeau/lien */

  memcpy( cmd + SCSI_OFF, cmdblk, sizeof(cmdblk) );

  /*
   * +------------------+
   * | struct sg_header | <- cmd
   * +------------------+
   * | copie de cmdblk  | <- cmd + SCSI_OFF
   * +------------------+
   */

  if (handle_scsi_cmd(sizeof(cmdblk), 0, cmd,
                      sizeof(Inqbuffer) - SCSI_OFF, Inqbuffer )) {
      fprintf( stderr, "Echec de la demande\n" );
      exit(2);
  }
  return (Inqbuffer + SCSI_OFF);
}

#define TESTUNITREADY_CMD 0
#define TESTUNITREADY_CMDLEN 6

#define ADD_SENSECODE 12
#define ADD_SC_QUALIFIER 13
#define NO_MEDIA_SC 0x3a
#define NO_MEDIA_SCQ 0x00
int TestForMedium ( void )
{
  /* demande de l'etat READY */
  static unsigned char cmdblk [TESTUNITREADY_CMDLEN] = {
      TESTUNITREADY_CMD, /* commande */
                      0, /* lun/reserve */
                      0, /* reserve */
                      0, /* reserve */
                      0, /* reserve */
                      0};/* reserve */
```

```
  memcpy( cmd + SCSI_OFF, cmdblk, sizeof(cmdblk) );

  /*
   * +------------------+
   * | struct sg_header | <- cmd
   * +------------------+
   * | copie de cmdblk  | <- cmd + SCSI_OFF
   * +------------------+
   */

  if (handle_scsi_cmd(sizeof(cmdblk), 0, cmd,
                          0, NULL)) {
      fprintf (stderr, "Echec du test d'unite prete\n");
      exit(2);
  }

  return
   *(((struct sg_header*)cmd)->sense_buffer +ADD_SENSECODE) !=
                                                    NO_MEDIA_SC ||
   *(((struct sg_header*)cmd)->sense_buffer +ADD_SC_QUALIFIER) !=
                                                    NO_MEDIA_SCQ;
}

void main( void )
{
  fd = open(DEVICE, O_RDWR);
  if (fd < 0) {
    fprintf( stderr, "Il faut les droits lecture/ecriture sur "DEVICE".\n" );
    exit(1);
  }

  /* on ecrit quelques champs du resultat de la requete */
  printf( "%s\n", Inquiry() + INQUIRY_VENDOR );

  /* on regarde si le medium est charge */
  if (!TestForMedium()) {
    printf("pas de medium charge\n");
  } else {
    printf("un medium est present\n");
  }
}
```