AIX version 7.2

Programmation Coherent Accelerator Processor Interface (CAPI)

AIX version 7.2

Programmation Coherent Accelerator Processor Interface (CAPI)

Important
Avant d'utiliser le présent document et le produit associé, prenez connaissance des informations générales figurant à la section «Remarques», à la page 31.

LE PRESENT DOCUMENT EST LIVRE EN L'ETAT SANS AUCUNE GARANTIE EXPLICITE OU IMPLICITE. IBM DECLINE NOTAMMENT TOUTE RESPONSABILITE RELATIVE A CES INFORMATIONS EN CAS DE CONTREFACON AINSI QU'EN CAS DE DEFAUT D'APTITUDE A L'EXECUTION D'UN TRAVAIL DONNE.

Ce document est mis à jour périodiquement. Chaque nouvelle édition inclut les mises à jour. Les informations qui y sont fournies sont susceptibles d'être modifiées avant que les produits décrits ne deviennent eux-mêmes disponibles. En outre, il peut contenir des informations ou des références concernant certains produits, logiciels ou services non annoncés dans ce pays. Cela ne signifie cependant pas qu'ils y seront annoncés.

Pour plus de détails, pour toute demande d'ordre technique, ou pour obtenir des exemplaires de documents IBM, référez-vous aux documents d'annonce disponibles dans votre pays, ou adressez-vous à votre partenaire commercial.

Vous pouvez également consulter les serveurs Internet suivants :

- http://www.fr.ibm.com (serveur IBM en France)
- http://www.ibm.com/ca/fr (serveur IBM au Canada)
- http://www.ibm.com (serveur IBM aux Etats-Unis)

Compagnie IBM France
Direction Qualité
17, avenue de l'Europe
92275 Bois-Colombes Cedex

La présente édition s'applique à AIX Version 7.2 et à toutes les éditions et modifications ultérieures, sauf indication contraire dans les nouvelles éditions.

Table des matières

Avis aux lecteurs canadiens

Le présent document a été traduit en France. Voici les principales différences et particularités dont vous devez tenir compte.

Illustrations

Les illustrations sont fournies à titre d'exemple. Certaines peuvent contenir des données propres à la France.

Terminologie

La terminologie des titres IBM peut différer d'un pays à l'autre. Reportez-vous au tableau ci-dessous, au besoin.

IBM France	**IBM Canada**
ingénieur commercial	représentant
agence commerciale	succursale
ingénieur technico-commercial	informaticien
inspecteur	technicien du matériel

Claviers

Les lettres sont disposées différemment : le clavier français est de type AZERTY, et le clavier français-canadien de type QWERTY.

OS/2 et Windows - Paramètres canadiens

Au Canada, on utilise :

- les pages de codes 850 (multilingue) et 863 (français-canadien),
- le code pays 002,
- le code clavier CF.

Nomenclature

Les touches présentées dans le tableau d'équivalence suivant sont libellées différemment selon qu'il s'agit du clavier de la France, du clavier du Canada ou du clavier des États-Unis. Reportez-vous à ce tableau pour faire correspondre les touches françaises figurant dans le présent document aux touches de votre clavier.

France	Canada	Etats-Unis
(Pos1)		Home
Fin	Fin	End
(PgAr)		PgUp
(PgAv)		PgDn
Inser	Inser	Ins
Suppr	Suppr	Del
Echap	Echap	Esc
Attn	Intrp	Break
Impr écran	ImpEc	PrtSc
Verr num	Num	Num Lock
Arrêt défil	Défil	Scroll Lock
(Verr maj)	FixMaj	Caps Lock
AltGr	AltCar	Alt (à droite)

Brevets

Il est possible qu'IBM détienne des brevets ou qu'elle ait déposé des demandes de brevets portant sur certains sujets abordés dans ce document. Le fait qu'IBM vous fournisse le présent document ne signifie pas qu'elle vous accorde un permis d'utilisation de ces brevets. Vous pouvez envoyer, par écrit, vos demandes de renseignements relatives aux permis d'utilisation au directeur général des relations commerciales d'IBM, 3600 Steeles Avenue East, Markham, Ontario, L3R 9Z7.

Assistance téléphonique

Si vous avez besoin d'assistance ou si vous voulez commander du matériel, des logiciels et des publications IBM, contactez IBM direct au 1 800 465-1234.

A propos du présent document

Vous pouvez utiliser Coherent Accelerator Processor Interface (CAPI) pour permettre à des accélérateurs FPGA (Field Programmable Gate Array) d'accéder à la mémoire (espace utilisateur) des applications directement.

Conventions typographiques

Le présent document utilise les conventions typographiques suivantes :

Gras	Identifie les commandes, les sous-programmes, les mots clés, les fichiers, les structures, les répertoires, ainsi que d'autres éléments dont le nom est défini par le système. La mise en caractères gras identifie également des objets graphiques, comme des boutons, des libellés et des icônes que vous sélectionnez.
Italique	Identifie les paramètres pour les noms ou valeurs réels que vous fournissez.
`Espacement fixe`	Identifie des exemples de valeurs de données spécifiques, des exemples de texte similaire à celui pouvant s'afficher, des exemples de portions de code de programme similaires à ce que vous pouvez écrire en tant que programmeur, des messages du système ou du texte que vous devez saisir.

Respect de la casse dans AIX

Dans le système d'exploitation AIX, tous les éléments sont sensibles à la casse, ce qui signifie que les minuscules et les majuscules sont différenciées. Vous pouvez, par exemple, utiliser la commande **ls** pour afficher la liste des fichiers. Si vous entrez `LS`, le système affiche un message d'erreur indiquant que la commande entrée `est introuvable`. De la même manière, **FILEA**, **FiLea** et **filea** sont trois noms de fichiers distincts, même s'ils se trouvent dans le même répertoire. Pour éviter toute action indésirable, vérifiez systématiquement que vous utilisez la casse appropriée.

ISO 9000

Les systèmes de gestion de la qualité utilisés pour le développement et la fabrication de ce produit sont en conformité avec les normes ISO 9000.

Programmation CAPI

Vous pouvez utiliser Coherent Accelerator Processor Interface (CAPI) pour permettre à des accélérateurs FPGA (Field Programmable Gate Array) d'accéder à la mémoire (espace utilisateur) des applications directement.

Les accélérateurs FPGA classiques effectuent des transferts via l'accès direct à la mémoire (DMA) dans une pile PCI (Peripheral Component Interconnect) afin de déplacer des données entre les accélérateurs et les applications. CAPI fournit une infrastructure à usage général possédant un accélérateur CAPI qui peut transférer des données de et vers la mémoire de l'application sans nécessiter d'accès direct à la mémoire.

Carte flash CAPI

Coherent Accelerator Processor Interface (CAPI) fournit une bande passante importante, un chemin à faible temps d'attente entre les unités externes, le coeur POWER8, et l'architecture de mémoire ouverte du système. Les cartes CAPI sont insérées dans les emplacements PCI Express (PCIe) x16 et utilisent la carte PCIe Gen3 comme mécanisme de transport sous-jacent.

Les unités compatibles avec CAPI peuvent remplacer des programmes d'application pouvant exécuter des programmes qui s'exécutent sur un coeur POWER8 ou fournir des implémentations d'accélération personnalisées. Les cartes flash CAPI éliminent la complexité du sous-système d'E-S de sorte qu'un accélérateur puisse intervenir comme partie intégrante d'une application. Le chemin du code est réduit car les applications peuvent interagir avec l'accélérateur flash directement, sans utiliser le noyau du système d'exploitation.

Bibliothèque de blocs flash CAPI

La bibliothèque de blocs pour la carte flash Coherent Accelerator Processor Interface (CAPI) fournit des interfaces d'espace utilisateur au disque flash CAPI au niveau du bloc ou du secteur et ignore le noyau pour les demandes d'E-S en lecture et en écriture. Elle crée une interface pour les applications de sorte que celles-ci n'aient pas besoin d'accéder aux détails de la carte flash CAPI de niveau inférieur.

Sur un système d'exploitation AIX, la bibliothèque de blocs pour la carte flash CAPI est `libcflsh_block.a`. Sur une plateforme Linux, il s'agit de `libcflsh_block.so`.

API cblk_init

Objectif

Initialise la bibliothèque de blocs pour la carte flash Coherent Accelerator Processor Interface (CAPI).

Syntaxe

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX

int rc = cblk_int(void *arg, int flags)
```

Description

L'API **cblk_init** initialise la bibliothèque de blocs pour la carte flash CAPI. Vous devez appeler l'API **cblk_init** avant d'utiliser une autre API dans la bibliothèque de blocs pour la carte flash CAPI.

Paramètres

arg
Ce paramètre n'est pas utilisé actuellement. Il a pour valeur NULL.

flags
Spécifie des indicateurs pour l'initialisation. La valeur par défaut est 0.

Valeurs de retour

0 L'API s'est exécutée sans erreur.

Valeur différente de zéro
Une erreur est survenue.

API cblk_term
Objectif

Nettoie les ressources pour la bibliothèque de blocs flash Coherent Accelerator Processor Interface (CAPI) lorsque la bibliothèque n'est plus utilisée.

Syntaxe

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX
int rc = cblk_term(void *arg, int flags)
```

Description

L'API cblk_term supprime la bibliothèque de blocs pour la carte flash CAPI lorsqu'elle n'est pas utilisée.

Paramètres

arg
Ce paramètre n'est pas utilisé actuellement (valeur NULL).

flags
Spécifie des indicateurs pour l'initialisation. La valeur par défaut est 0.

Valeurs de retour

0 L'API s'est exécutée sans erreur.

Valeur différente de zéro
Une erreur est survenue.

API cblk_open
Objectif

Ouvre un ensemble de blocs contigus qui composent un *fragment* sur une unité flash Coherent Accelerator Processor Interface (CAPI) pouvant effectuer des opérations d'E-S (lecture et écriture). Un fragment peut être considéré comme un numéro d'unité logique qui fournit l'accès aux secteurs 0 - *n*-1, où *n* est la taille du fragment dans les secteurs. Si des numéros d'unité logique sont spécifiés, le fragment est un sous-ensemble de secteurs sur un numéro d'unité logique physique.

Syntaxe

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX

chunk_id_t chunk_id = cblk_open(const char *path, int max_num_requests, int mode,
uint64_t ext_arg, int flags)
```

Description

L'API **cblk_open** crée un fragment sur un numéro d'unité logique flash CAPI. Ce fragment est utilisé pour les demandes d'E-S (cblk_read ou cblk_write). La valeur `chunk_id` renvoyée est affectée à un chemin

spécifique, depuis une carte spécifique au processus appelant. Les utilisateurs ne peuvent pas voir directement les secteurs physiques sous-jacents qui sont utilisés par un fragment de la couche blocs lorsque l'indicateur CBLK_OPN_VIRT_LUN est défini.

Lorsque l'appel API cblk_open aboutit, un ID de fragment qui représente l'instance de fragment créée est renvoyé au processus appelant utilisé pour les appels API ultérieurs.

Paramètres

path
Ce paramètre identifie le nom de fichier spécial pour le disque CAPI. Exemple : /dev/hdisk1 (AIX) et /dev/sg0 (Linux).

max_num_requests
Ce paramètre indique le nombre maximal de commandes pouvant être mises en file d'attente pour la carte pour un fragment spécifique à un moment précis. Si cette valeur est 0, la couche blocs choisit une taille par défaut. Si la valeur spécifiée est trop élevée, la demande cblk_open échoue avec la valeur d'erreur ENOMEM.

mode
Ce paramètre spécifie le mode d'accès (O_RDONLY, O_WRONLY ou O_RDWR).

ext_arg
Ce paramètre n'est pas utilisé actuellement.

flags
Ce paramètre correspond à l'ensemble des indicateurs de bit suivants :

CBLK_OPN_VIRT_LUN
Cet indicateur spécifie qu'un numéro d'unité logique virtuel est fourni sur un numéro d'unité logique physique. S'il n'est pas spécifié, l'accès direct au numéro d'unité logique physique complet est fourni. Cet indicateur est valable uniquement pour le stockage temporaire. Lorsque l'API cblk_close est appelée, tous les secteurs de données pour ce fragment sont libérés en vue de leur utilisation par d'autres opérations.

CBLK_OPN_NO_INTRP_THREADS
Cet indicateur spécifie que la bibliothèque de blocs cflash ne démarre pas d'unité d'exécution en arrière-plan pour le traitement et l'extraction des informations sur l'exécution asynchrone des demandes d'E-S depuis la carte CAPI. Le processus qui utilise cette bibliothèque doit appeler la bibliothèque cblk_aresult ou cblk_listio pour déterminer si les opérations d'E-S ont été exécutées.

CBLK_OPN_SCRUB_DATA
Cet indicateur est valide uniquement lorsque l'indicateur CBLK_OPN_VIRT_LUN est spécifié. Il spécifie que les données sur un numéro d'unité logique virtuel doivent être effacées pour que le numéro d'unité logique puisse être réutilisé par d'autres opérations. Il n'est pas pris en charge actuellement sur le système d'exploitation AIX.

CBLK_OPN_MPIO_FO
Cet indicateur est valide uniquement sur le système d'exploitation AIX. Il spécifie que la bibliothèque de blocs cflash utilise la reprise après incident MPIO (Multipath I/O). Un seul chemin est utilisé pour toutes les demandes d'E-S, sauf si des erreurs propres au chemin sont rencontrées. Si des erreurs de chemin surviennent, un chemin alternatif est utilisé, s'il en existe un. Afin d'identifier les chemins pour un disque flash CAPI, exécutez la commande **lspath -l hdiskN**. Cet indicateur n'est pas valide si l'indicateur CBLK_OPN_VIRT_LUN, CBLK_OPN_RESERVE ou CBLK_OPN_FORCED_RESERVE est spécifié.

CBLK_OPN_RESERVE
Cet indicateur est valide uniquement sur le système d'exploitation AIX. Cet indicateur spécifie que la bibliothèque de blocs cflash utilise un attribut de stratégie de réservation qui est associé au disque établissant les réservations de disque. Vous ne pouvez pas l'utiliser avec l'indicateur CBLK_OPN_MPIO_FO.

CBLK_OPN_FORCED_RESERVE
Cet indicateur est valide uniquement sur le système d'exploitation AIX. Le comportement de cet indicateur est identique à celui de l'indicateur CBLK_OPEN_RESERVE, à l'exception suivante : lorsque l'unité est ouverte pour la première fois, elle rompt toute réservation de disque non résolue. Vous ne pouvez pas utiliser cet indicateur avec l'indicateur CBLK_OPN_MPIO_FO.

Valeurs de retour

NULL_CHUNK_ID
Une erreur s'est produite.

API cblk_close
Objectif

Ferme un ensemble de blocs contigus appelés fragment sur une unité de stockage flash Coherent Accelerator Processor Interface (CAPI) pouvant exécuter des opérations d'E-S (lecture et écriture).

Syntaxe

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX
int rc = cblk_close(chunk_id_t chunk_id, int flags))
```

Description

L'API cblk_close libère les blocs qui sont associés à un fragment pour qu'ils puissent être réutilisés par d'autres opérations. Pour que les blocs puissent être réutilisés par d'autres opérations, les blocs de données doivent être effacés afin de supprimer toute donnée utilisateur si l'indicateur CBLK_OPN_SCRUB_DATA a été défini dans l'API cblk_open correspondante qui a renvoyé cette valeur chunk_id.

Paramètres

chunk_id
Descripteur du fragment en cours de fermeture et de libération en vue de sa réutilisation.

flags
Ensemble d'indicateurs de bit.

Valeurs de retour

0 L'API s'est exécutée sans erreur.

Valeur différente de zéro
Une erreur est survenue.

API cblk_get_lun_size
Objectif

Renvoie la taille (nombre de blocs) du numéro d'unité logique physique auquel un fragment spécifique est associé.

Syntaxe

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX
int rc = cblk_get_lun_size(chunk_id_t chunk_id, size_t *size, int flags))
```

Description

L'API cblk_get_lun_size renvoie le nombre de blocs du numéro d'unité logique physique qui est associé à ce fragment. Pour pouvoir utiliser le service cblk_get_lun_size, vous devez avoir exécuté l'API cblk_open afin de recevoir une valeur chunk_id valide.

Paramètres

chunk_id
Descripteur du fragment pour lequel la taille du numéro d'unité logique physique doit être renvoyée.

size
Spécifie le nombre total de blocs de 4K pour le numéro d'unité logique physique qui est associé à un fragment spécifique.

flags
Ensemble d'indicateurs de bit.

Valeurs de retour

0 L'API s'est exécutée sans erreur.

>0 Une erreur est survenue.

API cblk_get_size
Objectif

Renvoie la taille (nombre de blocs) affectée à un ID de fragment spécifique, qui est un numéro d'unité logique virtuel. En d'autres termes, l'indicateur `CBLK_OPN_VIRT_LUN` est spécifié pour l'appel cblk_open qui a renvoyé cet ID de fragment. Ce service ne fonctionne pas pour les numéros d'unité logique pour lesquels l'indicateur `CBLK_OPN_VIRT_LUN` n'a pas été défini lorsque les fragments ont été ouverts avec l'API cblk_open.

Syntaxe

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX
int rc = cblk_get_size(chunk_id_t chunk_id, size_t *size, int flags))
```

Description

Le service cblk_get_size renvoie le nombre de blocs alloués à un fragment spécifique. Pour pouvoir utiliser le service cblk_get_size, vous devez avoir exécuté l'API cblk_open afin de recevoir une valeur `chunk_id` valide.

Paramètres

chunk_id
Descripteur du fragment pour lequel la taille du numéro d'unité logique doit être changée.

size
Spécifie le nombre de blocs de 4K pour le numéro d'unité logique qui est associé à un fragment spécifique.

flags
Ensemble d'indicateurs de bit.

Valeurs de retour

0 L'API s'est exécutée sans erreur.

>0 Une erreur est survenue.

API cblk_set_size
Objectif

Affecte une taille (nombre de blocs) à un ID de fragment spécifique qui est un numéro d'unité logique virtuel. En d'autres termes, l'indicateur `CBLK_OPN_VIRT_LUN` est spécifié pour l'appel cblk_open qui a renvoyé cet ID de fragment. Si les blocs sont déjà affectés à cet ID de fragment, vous pouvez augmenter

ou réduire la taille en spécifiant une taille plus grande ou plus petite. Ce service ne fonctionne pas pour les numéros d'unité logique pour lesquels l'indicateur CBLK_OPN_VIRT_LUN n'a pas été défini lorsque les fragments ont été ouverts avec l'API cblk_open.

Syntaxe

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX
int rc = cblk_set_size(chunk_id_t chunk_id, size_t size, int flags))
```

Description

Lorsque vous utilisez les numéros d'unité logique virtuels, le service cblk_set_size alloue un nombre de blocs à un fragment spécifique. L'API cblk_set_size doit être appelée avant que cblk_read ou cblk_write n'appelle ce fragment. Pour pouvoir utiliser le service cblk_set_size et recevoir une valeur chunk_id valide, vous devez effectuer l'appel cblk_open.

Si des blocs ont été initialement affectés à ce fragment et n'ont pas été réutilisés après l'allocation par l'API cblk_set_size de nouveaux blocs au même fragment, et si l'indicateur CBLK_SCRUB_DATA_FLG est défini dans le paramètre **flags**, les blocs d'origine sont nettoyés avant que d'autres opérations cblk_set_size ne puissent les réutiliser.

En cas de réussite de l'API cblk_set_size, le fragment peut avoir une taille d'adresse de bloc logique, comprise entre 0 et 1, qui peut être lue ou écrite.

Paramètres

chunk_id
Descripteur du fragment pour lequel la taille du numéro d'unité logique doit être définie.

size
Spécifie le nombre de blocs de 4K pour le numéro d'unité logique qui est associé à un fragment spécifique.

flags
Ensemble d'indicateurs de bit.

Valeurs de retour

0 L'API s'est exécutée sans erreur.

>0 Une erreur est survenue.

API cblk_get_stats
Objectif

Renvoie des statistiques pour un ID de fragment spécifique.

Syntaxe

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX
typedef struct chunk_stats_s {
uint64_t max_transfer_size;    /* Taille maximale de transfert en */
                               /* blocs de ce fragment. */
uint64_t num_reads;            /* Nombre total de lectures émises */
                               /* via l'interface cblk_read */
uint64_t num_writes;           /* Nombre total d'écritures émises */
                               /* via l'interface cblk_write */
uint64_t num_areads;           /* Nombre total de lectures asynchrones */
                               /* émises via l'interface cblk_aread */
uint64_t num_awrites;          /* Nombre total d'écritures asynchrones */
                               /* émises via l'interface cblk_awrite */
uint32_t num_act_reads;        /* Nombre en cours de lectures actives */
                               /* via l'interface cblk_read */
```

```
uint32_t num_act_writes;       /* Nombre en cours d'écritures actives */
                               /* via l'interface cblk_write */
uint32_t num_act_areads;       /* Nombre en cours de lectures asynchrones */
                               /* actives via l'interface cblk_aread */
uint32_t num_act_awrites;      /* Nombre en cours d'écritures asynchrones */
                               /* actives via l'interface cblk_awrite */
uint32_t max_num_act_writes;   /* Seuil supérieur pour le nombre maximal */
                               /* d'écritures actives en une fois */
uint32_t max_num_act_reads;    /* Seuil supérieur pour le nombre maximal */
                               /* de lectures actives en une fois */
uint32_t max_num_act_awrites;  /* Seuil supérieur pour le nombre maximal */
                               /* d'écritures asynchrones actives */
                               /* en une fois. */
uint32_t max_num_act_areads;   /* Seuil supérieur pour le nombre maximal */
                               /* de lectures asynchrones actives */
                               /* en une fois. */
uint64_t num_blocks_read;      /* Nombre total de lectures de blocs */
uint64_t num_blocks_written;   /* Nombre total d'écritures de blocs */
uint64_t num_errors;           /* Nombre total de réponses d'erreur */
                               /* affichées */
uint64_t num_aresult_no_cmplt; /* Nombre de retours de cblk_aresult */
                               /* sans achèvement de */
                               /* commande */
uint64_t num_retries;          /* Nombre total de tentatives */
                               /* d'exécution de commande. */
uint64_t num_timeouts;         /* Nombre total d'expirations */
                               /* de commande. */
uint64_t num_fail_timeouts;    /* Nombre total d'expirations */
                               /* de commande ayant entraîné */
                               /* l'échec d'une commande. */
uint64_t num_no_cmds_free;     /* Nombre total de fois où une */
                               /* commande libre n'était pas disponible */
uint64_t num_no_cmd_room ;     /* Nombre total de fois où l'espace */
                               /* était insuffisant pour émettre une commande */
                               /* pour AFU. */
uint64_t num_no_cmds_free_fail; /* Nombre total de fois où une */
                               /* commande libre n'était pas disponible, ce */
                               /* qui a entraîné l'échec d'une demande */
uint64_t num_fc_errors;        /* Nombre total de réponses d'erreur */
                               /* FC affichées */
uint64_t num_port0_linkdowns;  /* Nombre total de déconnexions */
                               /* affichées sur le port 0. */
uint64_t num_port1_linkdowns;  /* Nombre total de déconnexions */
                               /* affichées sur le port 1. */
uint64_t num_port0_no_logins;  /* Nombre total de toutes les opérations de */
                               /* connexion non effectuées affichées sur le port 0. */
uint64_t num_port1_no_logins;  /* Nombre total de toutes les opérations de */
                               /* connexion non effectuées affichées sur le port 1. */
uint64_t num_port0_fc_errors;  /* Nombre total d'erreurs FC générales */
                               /* affichées sur le port 0. */
uint64_t num_port1_fc_errors;  /* Nombre total d'erreurs FC générales */
                               /* affichées sur le port 1. */
uint64_t num_cc_errors;        /* Nombre total de réponses de */
                               /* condition de vérification affichées */
uint64_t num_afu_errors;       /* Nombre total de réponses d'erreur */
                               /* AFU affichées */
uint64_t num_capi_false_reads; /* Nombre total de fois où */
                               /* l'interrogation a indiqué qu'une lecture était prête */
                               /* alors qu'il n'y avait pas de données à lire. */
uint64_t num_capi_adap_resets; /* Nombre total d'erreurs de réinitialisation */
                               /* de carte. */
uint64_t num_capi_afu_errors;  /* Nombre total de réponses d'erreur */
                               /* CAPI affichées */
uint64_t num_capi_afu_intrpts; /* Nombre total d'interruptions */
                               /* AFU CAPI pour les réponses */
                               /* de commande affichées. */
uint64_t num_capi_unexp_afu_intrpts; /* Nombre total d'interruptions */
```

```
                              /* AFU inattendues */
uint64_t num_active_threads;  /* Nombre en cours d'unités */
                              /* d'exécution en cours. */
uint64_t max_num_act_threads; /* Nombre maximal d'unités */
                              /* d'exécution en cours simultanément. */
uint64_t num_cache_hits;      /* Nombre total de réussites en cache */
                              /* affichées pour toutes les lectures */
} chunk_stats_t;
int rc = cblk_get_stats(chunk_id_t chunk_id, chunk_stats_t *stats, int flags))
```

Description

Le service cblk_get_stats renvoie des statistiques pour un ID de fragment spécifique.

Paramètres

chunk_id
Descripteur du fragment pour lequel les statistiques doivent être déterminées.

stats
Spécifie l'adresse de la structure `chunk_stats_t`.

flags
Ensemble d'indicateurs de bit.

Valeurs de retour

0 L'API s'est exécutée sans erreur.

>0 Une erreur est survenue.

API cblk_read
Objectif

Lit des blocs de 4K depuis le fragment à l'adresse de bloc logique spécifiée dans le tampon spécifié. Lorsque vous utilisez des numéros d'unité logique virtuels, cette adresse de bloc logique n'est pas la même que celle du numéro d'unité logique car le fragment ne commence pas toujours à l'adresse de bloc logique 0 du numéro d'unité logique.

Syntaxe

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX
int rc = cblk_read(chunk_id_t chunk_id, void *buf, off_t lba, size_t nblocks, int flags));
```

Description

Le service cblk_read lit des données depuis le fragment et les place dans le tampon fourni. Cet appel est bloqué jusqu'à ce que l'opération de lecture aboutisse avec ou sans erreur. Cela signifie qu'il ne s'arrête pas et que l'appel suivant n'est pas effectué tant que l'opération de lecture n'est pas terminée. Dans le cas d'un numéro d'unité logique virtuel, vous devez appeler l'API cblk_set_size avant d'appeler l'API cblk_read, cblk_write, cblk_aread ou cblk_awrite pour un fragment spécifique.

Paramètres

chunk_id
Descripteur du fragment en cours de lecture.

buf
Spécifie le tampon dans lequel les données sont lues depuis le fragment. Cette valeur de paramètre doit respecter des découpes de 16 octets.

lba
Spécifie l'adresse de bloc logique (décalage de 4K) dans le fragment.

nblocks
Spécifie la taille du transfert en secteurs de 4K. Pour un numéro d'unité logique physique, la taille maximale est de 16 Mo. Pour un numéro d'unité logique virtuel, la taille maximale est de 4K.

flags
Ensemble d'indicateurs de bit.

Valeurs de retour

-1 Indique une erreur. Un numéro d'erreur est défini pour plus de détails.

0 Indique qu'aucune donnée n'a été lue.

***n* > 0**
Indique que l'opération de lecture a réussi. *n* est le nombre de blocs lus.

API cblk_write
Objectif

Ecrit des blocs de 4K dans le fragment à l'adresse de bloc logique spécifiée en utilisant les données provenant du tampon spécifié. Lorsque vous utilisez des numéros d'unité logique virtuels, cette adresse de bloc logique n'est pas la même que celle du numéro d'unité logique car le fragment ne commence pas à l'adresse de bloc logique 0.

Syntaxe

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX
int rc = cblk_write(chunk_id_t chunk_id, void *buf, off_t lba, size_t nblocks, int flags));
```

Description

L'API cblk_write écrit des données depuis le tampon spécifié dans le fragment à l'adresse de bloc logique indiquée. L'appel cblk_write est bloqué jusqu'à ce que l'opération d'écriture aboutisse avec ou sans erreur. Cela signifie qu'il ne s'arrête pas et que l'appel suivant n'est pas effectué tant que l'opération d'écriture n'est pas terminée. Dans le cas d'un numéro d'unité logique virtuel, vous devez appeler l'API cblk_set_size avant d'appeler l'API cblk_write pour un fragment spécifique.

Paramètres

chunk_id
Descripteur du fragment en cours d'écriture.

buf
Spécifie le tampon des données écrites dans le fragment. Cette valeur de paramètre doit respecter des découpes de 16 octets.

lba
Spécifie l'adresse de bloc logique (décalage de 4K) dans le fragment.

nblocks
Spécifie la taille du transfert en secteurs de 4K. Pour un numéro d'unité logique physique, la taille maximale est de 16 Mo. Pour un numéro d'unité logique virtuel, la taille maximale est de 4K.

flags
Ensemble d'indicateurs de bit.

Valeurs de retour

-1 Indique une erreur. Un numéro d'erreur est défini pour plus de détails.

0 Indique qu'aucune donnée n'a été écrite.

***n* > 0**
Indique que l'opération d'écriture a réussi. *n* est le nombre de blocs écrits.

API cblk_aread
Objectif

Lit des blocs de 4K depuis le fragment à l'adresse de bloc logique spécifiée dans le tampon spécifié. Lorsque vous utilisez des numéros d'unité logique virtuels, cette adresse de bloc logique n'est pas la même que celle du numéro d'unité logique car le fragment ne commence pas à l'adresse de bloc logique 0.

Syntaxe

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX


typedef enum {
    CBLK_ARW_STATUS_PENDING = 0,     /* La commande n'a pas abouti */
    CBLK_ARW_STATUS_SUCCESS = 1      /* La commande s'est terminée sans erreur */
    CBLK_ARW_STATUS_INVALID = 2      /* La demande de l'appelant n'est pas valide */
    CBLK_ARW_STATUS_FAIL = 3         /* La commande s'est terminée avec une erreur */
} cblk_status_type_t;

typedef struct cblk_arw_status_s {
    cblk_status_type_t status;       /* Statut de la commande */
                                     /* Voir la zone errno pour des détails */
                                     /* supplémentaires sur l'incident */
    size_t blocks_transferred;       /* Nombre de blocs transférés par */
                                     /* cette demande. */
    int errno;                       /* Numéro d'erreur lorsque le statut */
                                     /* indique CBLK_ARW_STAT_FAIL */
} cblk_arw_status_t;

int rc = cblk_aread(chunk_id_t chunk_id, void *buf, off_t lba, size_t nblocks, int
*tag, cblk_arw_status_t *status, int flags));
```

Description

Le service cblk_aread lit des données depuis le fragment et les place dans le tampon fourni. Cet appel n'est pas bloqué tant que l'opération de lecture n'est pas terminée. Cela signifie que cet appel s'arrête et que l'appel suivant est effectué immédiatement après l'émission de la demande, avant la fin de l'opération de lecture. Ensuite, un appel cblk_aresult doit être effectué pour déterminer l'achèvement. Dans le cas d'un numéro d'unité logique virtuel, vous devez appeler l'API cblk_set_size avant d'appeler l'API cblk_aread.

Paramètres

chunk_id
Descripteur du fragment en cours de lecture.

buf
Spécifie le tampon dans lequel les données sont lues depuis le fragment. Cette valeur de paramètre doit respecter des découpes de 16 octets.

lba
Spécifie l'adresse de bloc logique (décalage de 4K) dans le fragment.

nblocks
Spécifie la taille du transfert en secteurs de 4K. Pour un numéro d'unité logique physique, la taille maximale est de 16 Mo. Pour un numéro d'unité logique virtuel, la taille maximale est de 4K.

tag
Spécifie l'identificateur renvoyé qui permet d'identifier de façon unique chaque commande émise.

status
Spécifie l'adresse fournie par le processus appelant, que la bibliothèque `capiblock` met à jour lorsque l'exécution de l'API cblk_aread est terminée. Les applications peuvent utiliser le processus d'interrogation pour l'argument **status** au lieu du service cblk_aresult.

La carte CAPI ne peut pas mettre à jour cette zone directement. Des unités d'exécution logicielles sont requises pour mettre à jour le paramètre `status`. Cette zone n'est pas utilisée si l'indicateur `CBLK_OPN_NO_INTRP_THREADS` a été spécifié pour l'API cblk_open qui a renvoyé cette valeur `chunk_id`.

flags
Ensemble des indicateurs de bit suivants :

CBLK_ARW_WAIT_CMD_FLAGS
Bloque le service cblk_aread jusqu'à ce qu'une commande libre soit disponible pour émettre la demande. Sinon, ce service peut renvoyer la valeur -1 avec la valeur d'erreur EWOULDBLOCK (si aucune commande libre n'est disponible).

CBLK_ARW_USER_TAG_FLAGS
Indique que le processus appelant spécifie une balise définie par l'utilisateur pour cette demande. Ensuite, l'appelant doit utiliser cette balise avec l'API cblk_aresult et définir son indicateur `CBLK_ARESULT_USER_TAG`.

CBLK_ARW_USER_STATUS_FLAG
Indique que le processus appelant définit le paramètre **status** qui sera mis à jour une fois l'exécution de la commande terminée.

Valeurs de retour

-1 Indique une erreur. Un numéro d'erreur est défini pour plus de détails.

0 Indique que cette API a été exécutée sans erreur.

***n* > 0**
Indique que l'opération de lecture est terminée (éventuellement depuis le cache). *n* est le nombre de blocs lus.

API cblk_awrite
Objectif

Ecrit des blocs de 4K dans le fragment à l'adresse de bloc logique spécifiée en utilisant les données provenant du tampon spécifié. Lorsque vous utilisez des numéros d'unité logique virtuels, cette adresse de bloc logique n'est pas la même que celle du numéro d'unité logique car le fragment ne commence pas à l'adresse de bloc logique 0.

Syntaxe

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX

typedef enum {
       CBLK_ARW_STAT_NOT_ISSUED = 0  /* La commande n'a pas été émise */
       CBLK_ARW_STAT_PENDING = 1     /* La commande n'a pas abouti */
       CBLK_ARW_STAT_SUCCESS = 2     /* La commande s'est terminée sans erreur */
       CBLK_ARW_STAT_FAIL = 3        /* La commande s'est terminée avec une erreur */
} cblk_status_type_t;

typedef struct cblk_arw_status_s {
       cblk_status_type_t status;    /* Statut de la commande */
                                     /* Voir la zone errno pour des détails */
                                     /* supplémentaires sur l'incident */
       size_t blocks_transferred;    /* Nombre de blocs transférés par */
                                     /* cette demande. */
```

```
       int errno;                   /* Numéro d'erreur lorsque le statut */
                                    /* indique CBLK_ARW_STAT_FAIL */
} cblk_arw_status_t;

int rc = cblk_awrite(chunk_id_t chunk_id, void *buf, off_t lba, size_t nblocks, int
*tag, cblk_arw_status_t *status, int flags));
```

Description

L'API cblk_awrite écrit des données depuis le tampon spécifié dans le fragment à l'adresse de bloc logique indiquée. Cet appel n'est pas bloqué tant que l'opération d'écriture n'est pas terminée. Cela signifie que cet appel s'arrête et que l'appel suivant est effectué immédiatement après l'émission de la demande, avant la fin de l'opération d'écriture. Ensuite, un appel cblk_aresult doit être effectué pour déterminer l'achèvement. Dans le cas d'un numéro d'unité logique virtuel, vous devez appeler l'API cblk_set_size avant d'appeler l'API cblk_awrite.

Paramètres

chunk_id
Descripteur du fragment en cours d'écriture.

buf
Spécifie le tampon des données écrites dans le fragment. Cette valeur de paramètre doit respecter des découpes de 16 octets.

lba
Spécifie l'adresse de bloc logique (décalage de 4K) dans le fragment.

nblocks
Spécifie la taille du transfert en secteurs de 4K. Pour un numéro d'unité logique physique, la taille maximale est de 16 Mo. Pour un numéro d'unité logique virtuel, la taille maximale est de 4K.

tag
Spécifie l'identificateur renvoyé qui permet d'identifier de façon unique chaque commande émise.

status
Spécifie l'adresse fournie par le processus appelant, que la bibliothèque `capiblock` met à jour lorsque l'exécution de l'API cblk_aread est terminée. L'API cblk_aread peut être utilisée par une application à la place du service cblk_aresult.

La carte CAPI ne peut pas mettre à jour cette zone directement. Elle requiert des unités d'exécution logicielles pour mettre à jour la région de statut. Cette zone n'est pas utilisée si l'indicateur `CBLK_OPN_NO_INTRP_THREADS` a été spécifié pour l'API cblk_open qui a renvoyé cette valeur `chunk_id`.

flags
Ensemble des indicateurs de bit suivants :

CBLK_ARW_WAIT_CMD_FLAGS
Bloque le service cblk_aread jusqu'à ce qu'une commande libre soit disponible pour émettre la demande. Sinon, ce service peut renvoyer la valeur -1 avec la valeur d'erreur EWOULDBLOCK (si aucune commande libre n'est disponible).

CBLK_ARW_USER_TAG_FLAGS
Indique que le processus appelant spécifie une balise définie par l'utilisateur pour cette demande. Ensuite, le processus appelant doit utiliser cette balise avec l'API cblk_aresult et définir son indicateur `CBLK_ARESULT_USER_TAG`.

CBLK_ARW_USER_STATUS_FLAG
Indique que le processus appelant définit le paramètre **status** qui sera mis à jour une fois l'exécution de la commande terminée.

Valeurs de retour

-1 Indique une erreur. Un numéro d'erreur est défini pour plus de détails.

0 Indique que cette API a été émise correctement.

***n* > 0**
Indique que l'opération de lecture est terminée. *n* est le nombre de blocs écrits.

API cblk_aresult
Objectif

Renvoie des informations de statut et d'achèvement pour des demandes asynchrones.

Syntaxe

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX
rc = cblk_aresult(chunk_id_t chunk_id, int *tag, uint64_t *status, int flags);
```

Description

L'API cblk_aresult renvoie le statut des demandes en attente qui sont émises avec l'API cblk_aread ou cblk_awrite. Si ces demandes en attente sont terminées, cette API renvoie les informations d'achèvement.

Paramètres

chunk_id
Descripteur du fragment en cours d'écriture.

tag
Pointeur balisant le processus appelant qui attend l'achèvement de la demande. Si l'indicateur CBLK_ARESULT_NEXT_TAG est défini, cette zone renvoie la balise pour l'achèvement de la demande asynchrone suivante.

status
Pointeur du statut. Le statut est renvoyé lorsqu'une demande est terminée.

flags
Spécifie les indicateurs suivants pour l'API cblk_aresult :

CBLK_ARESULT_BLOCKING
Spécifiez cet indicateur si vous voulez que l'API cblk_aresult soit bloquée jusqu'à la fin d'une commande (à condition que des commandes actives existent). Si l'indicateur CBLK_ARESULT_NEXT_TAG est spécifié, cet appel s'arrête et l'appel suivant est effectué dès qu'une demande d'E-S asynchrone est terminée.

CBLK_ARESULT_USER_TAG
Utilisez cet indicateur pour vérifier le statut d'une demande asynchrone qui a été émise avec une balise spécifiée par l'utilisateur.

Valeurs de retour

-1 Indique une erreur. Un numéro d'erreur est défini pour plus de détails.

0 Indique que cette API a été émise correctement.

***n* > 0**
Indique que la demande est terminée. *n* est le nombre de blocs lus ou écrits.

API cblk_clone_after_fork
Objectif

Désigne un processus enfant devant accéder au même numéro d'unité logique virtuel que le processus parent. Ce service fonctionne pour la plateforme Linux seulement.

Syntaxe

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX
rc = cblk_clone_after_fork(chunk_id_t chunk_id, int mode, int flags);
```

Description

Le service cblk_clone_after_fork désigne un processus enfant devant accéder aux données du processus parent. Le processus enfant doit effectuer cette opération immédiatement après l'appel système **fork()** en utilisant l'ID de fragment du parent pour accéder à ce stockage. Si le processus enfant n'effectue pas cette opération, il n'aura pas accès aux ID de fragment du parent. Ce service ne fonctionne pas pour les numéros d'unité logique physiques.

Remarque : ce service fonctionne pour la plateforme Linux seulement.

Paramètres

chunk_id
Descripteur du fragment utilisé par le processus parent. Si cet appel s'arrête et que l'appel suivant est effectué, cet ID de fragment peut aussi être utilisé par le processus enfant.

mode
Spécifie le mode d'accès pour le processus enfant (O_RDONLY, O_WRONLY ou O_RDWR).

Remarque : les processus enfant ne peuvent pas disposer d'un accès plus large que le processus parent. Les processus descendants peuvent disposer d'un accès moindre.

flags
Ce paramètre est un indicateur de bit qui est spécifié par le processus appelant.

Valeurs de retour

0 Indique que la demande a abouti.

-1 Indique une erreur. Un numéro d'erreur est défini pour plus de détails.

API cblk_listio
Objectif

Envoie plusieurs demandes d'E-S au disque flash Coherent Accelerator Processor Interface (CAPI) en un seul appel et attend l'achèvement de plusieurs demandes d'E-S depuis un disque flash CAPI.

Syntaxe

```
#include <capiblock.h> for Linux or <sys/capiblock.h> for AIX

typedef struct cblk_io {
      uchar version;                /* Version de la structure */
#define CBLK_IO_VERSION_0 "I"       /* Version initiale 0 */
      int flags;                    /* Indicateurs pour la demande */
#define CBLK_IO_USER_TAG 0x0001     /* L'appelant spécifie une balise */
                                    /* définie par l'utilisateur. */
#define CBLK_IO_USER_STATUS 0x0002  /* L'appelant spécifie une emplacement de */
                                    /* statut à mettre à jour */
#define CBLK_IO_PRIORITY_REQ 0x0004 /* Demande à priorité (élevée) qui */
                                    /* doit être envoyée via des demandes */
```

```
                                    /* sans priorité */
      uchar request_type;           /* Type de demande */
#define CBLK_IO_TYPE_READ 0x01      /* Demande de lecture de données */
#define CBLK_IO_TYPE_WRITE 0x02     /* Demande d'écriture de données */
      void *buf;                    /* Tampon de données pour la demande */
      offset_t lba;                 /* Adresse de bloc logique de départ pour */
                                    /* la demande. */
      size_t nblocks;               /* Taille de la demande en fonction du */
                                    /* nombre de blocs */
      int tag;                      /* Balise pour la demande */
      cblk_arw_status_t stat;       /* Statut de la demande */
} cblk_io_t

int rc = cblk_listio(chunk_id_t chunk_id,cblk_io_t
*issue_io_list[],int
issue_io_items,cblk_io_t
*pending_io_list[],int
pending_io_items, cblk_io_t
*wait_io_list[], int wait_items,
cblk_io_t *completion_io_list[],int
*completion_items, uint64_t timeout, int
flags));
```

Description

Le service cblk_listio fournit une interface permettant d'envoyer plusieurs demandes d'E-S en un seul appel et de déterminer si plusieurs demandes d'E-S sont terminées en un seul appel. Les demandes individuelles sont spécifiées par le type cblk_io_t, qui inclut un tampon de données, une adresse de bloc logique de départ, et une taille de transfert en blocs de 4K.

Ce service peut mettre à jour les demandes d'E-S qui sont associées au type cblk_io_t (c'est-à-dire le statut de mise à jour, les balises et les indicateurs en fonction de la disposition de la demande d'E-S).

Ce service ne peut pas être utilisé pour vérifier l'achèvement des demandes d'E-S émises via l'API cblk_aread ou cblk_awrite.

Paramètres

chunk_id
Descripteur du fragment associé aux demandes d'E-S.

issue_io_list
Ce paramètre spécifie un tableau de demandes d'E-S à envoyer à des disques flash CAPI. Chaque élément de tableau individuel de type `cblk_io_t` spécifie une demande d'E-S individuelle contenant le tampon de données, l'adresse de bloc logique de départ et une taille de transfert en blocs de 4K. Ces éléments de tableau peuvent être mis à jour par cette API pour indiquer le statut d'achèvement et des balises. La zone de statut des éléments de tableau `cblk_io_t` individuels est initialisée par cette API. Si le paramètre **issue_io_list** est nul, cette API peut être utilisée pour attendre l'achèvement d'autres demandes qui ont été émises par les appels cblk_listio précédents en définissant le paramètre **pending_io_list**.

issue_io_items
Spécifie le nombre d'éléments de tableau dans le tableau **issue_io_list**.

pending_io_list
Spécifie un tableau de demandes d'E-S qui ont été émises via une demande cblk_listio précédente. Vous pouvez utiliser le paramètre **pending_io_list** pour déterminer si la demande d'E-S est terminée, sans attendre l'achèvement de toutes les demandes (en définissant le paramètre **completion_io_list**).

pending_io_items
Spécifie le nombre d'éléments de tableau dans le tableau **pending_io_list**.

wait_io_list
Spécifie le tableau des demandes d'E-S pour lequel le service cblk_listio est bloqué jusqu'à l'achèvement des demandes d'E-S. Ces demandes d'E-S doivent aussi être spécifiées dans le paramètre **issue_io_list** ou le paramètre **pending_io_list**. Si une demande d'E-S dans le tableau **issue_io_list** ne peut pas être émise par le processus appelant en raison de paramètres non valides ou car les ressources nécessaires ne sont pas disponibles, les éléments de cette demande d'E-S dans `io_list` sont mis à jour pour indiquer l'échec (le statut devient CBLK_ARW_STAT_NOT_ISSUED) et l'API cblk_listio n'attend pas l'achèvement de cette demande d'E-S. Ainsi, toutes les demandes d'E-S dans le tableau **wait_io_list** qui sont terminées ont le statut CBLK_ARW_STAT_SUCCESS ou CBLK_ARW_STAT_FAIL. Le statut n'est pas mis à jour pour les demandes d'E-S qui ne sont pas terminées.

wait_items
Spécifie le nombre d'éléments de tableau dans le tableau **wait_io_list**.

completion_io_list
Ce paramètre est associé par le processus appelant à un tableau initialisé (à zéro) de demandes d'E-S et le paramètre **completion_items** a pour valeur le nombre d'éléments de tableau dans le tableau. Au retour de l'API cblk_listio, le tableau contient les demandes d'E-S qui sont spécifiées dans les paramètres **issue_io_list** et **pending_io_list** et qui ont été terminées par l'unité CAPI, mais qui ne sont pas spécifiées dans le paramètre **wait_io_list**. Si une demande d'E-S dans le tableau `io_list` ne peut pas être émise par le processus appelant en raison de paramètres non valides ou car les ressources nécessaires ne sont pas disponibles, l'élément de cette demande d'E-S n'est pas copié dans le paramètre **completion_io_list** et son statut dans le tableau `io_list` est mis à jour pour indiquer l'échec (le statut devient CBLK_ARW_STAT_NOT_ISSUED). Ainsi, toutes les demandes d'E-S qui sont renvoyées dans cette liste ont le statut CBLK_ARW_STAT_SUCCESS ou CBLK_ARW_STAT_FAIL.

completion_items
Ce paramètre est associé par le processus appelant à l'adresse du nombre d'éléments de tableau que cette API a placés dans le paramètre **completion_io_list**. Au retour de cette API, la valeur de ce paramètre est mis à jour avec le nombre de demandes d'E-S placées dans le paramètre **completion_io_list**.

timeout
Spécifie la durée en microsecondes pendant laquelle attendre l'achèvement de toutes les demandes d'E-S qui sont indiquées dans le paramètre **wait_io_list**. Ce paramètre n'est valide que si le paramètre **wait_io_list** n'est pas nul. Si l'une des demandes d'E-S dans le paramètre **wait_io_list** ne se termine pas au cours de cette période, cette API renvoie la valeur -1 et définit un numéro d'erreur de type ETIMEDOUT (lorsque cette erreur survient, il se peut que certaines commandes aient été exécutées dans le paramètre **wait_io_list**). Ainsi, le processus appelant doit vérifier chaque demande se trouvant dans le paramètre **wait_io_list** afin de déterminer quelles sont celles qui sont terminées. Il doit supprimer les éléments terminés du paramètre **pending_io_list** avant l'appel suivant de cette API. La valeur de délai 0 indique que cette API sera bloquée tant que les demandes qui se trouvent dans le paramètre **wait_io_list** ne seront pas terminées.

flags
Spécifie l'indicateur de bit suivant :

CBLK_LISTIO_WAIT_ISSUE_CMD
Bloque l'API cblk_listio jusqu'à ce qu'une commande libre soit disponible pour émettre toutes les demandes même si la valeur de délai est dépassée et le paramètre `CBLK_LISTIO_WAIT_CMD_FLAG` est défini. Sinon, ce service peut renvoyer la valeur -1 avec une erreur de type EWOULDBLOCK si aucune commande libre n'est disponible (dans ce cas, il se peut que certaines commandes aient été placées en file d'attente pour la liste émise. Le processus appelant doit examiner les demandes d'E-S individuelles qui se trouvent dans le paramètre **issue_io_list** afin de déterminer quelles sont celles qui ont échoué.)

Valeurs de retour

-1 Une erreur et un numéro d'erreur sont définis pour plus de détails.

0 L'exécution de cette API s'est terminée sans erreur.

Bibliothèque de combinaisons clé-valeur flash CAPI

La bibliothèque de combinaisons clé-valeur fournit une interface aux unités flash Coherent Accelerator Processor Interface (CAPI) pour le stockage, l'extraction et la gestion des tableaux. Elle mappe la sémantique clé-valeur à la bibliothèque de blocs flash CAPI.

Sur un système d'exploitation AIX, la bibliothèque de combinaisons clé-valeur est `libarkdb.a`. Sur une plateforme Linux, il s'agit de `libarkdb.so`.

API ark_create
Objectif

Crée une instance de magasin de combinaisons clé-valeur.

Syntaxe

```
int ark_create(path, ark, flags)
char * file;
ARK ** handle;
uint64_t flags;
```

Description

L'API ark_create crée une instance de magasin de combinaisons clé-valeur sur le système hôte.

Le paramètre **path** peut être utilisé pour spécifier le fichier spécial (par exemple le fichier `/dev/sdx` pour la plateforme Linux ou le fichier `/dev/hdiskx` pour le système d'exploitation AIX) qui représente le numéro d'unité logique physique créé sur le stockage flash. Si le paramètre **path** n'est pas un fichier spécial, l'API suppose que le fichier doit être utilisé pour le magasin de combinaisons clé-valeur. Si le fichier n'existe pas, il est créé. Si le paramètre **path** a pour valeur NULL, la mémoire est utilisée pour le magasin de combinaisons clé-valeur.

Le paramètre **flags** indique les propriétés du magasin de combinaisons clé-valeur. Si vous voulez indiquer un fichier spécial pour le numéro d'unité logique physique, vous pouvez spécifier l'utilisation d'un magasin de combinaisons clé-valeur existant ou la création du magasin de combinaisons clé-valeur sur un numéro d'unité logique virtuel. Par défaut, le numéro d'unité logique physique entier est utilisé pour le magasin de combinaisons clé-valeur. Si un numéro d'unité logique virtuel est requis, l'indicateur de bit `ARK_KV_VIRTUAL_LUN` doit être défini dans le paramètre **flags**.

Un magasin de combinaisons clé-valeur qui est configuré pour l'utilisation du numéro d'unité logique physique entier peut être conservé. Vous pouvez arrêter une instance de magasin de combinaisons clé-valeur en utilisant la persistance (c'est-à-dire en sauvegardant l'état en cours des données) d'un magasin de combinaisons clé-valeur, puis vous pouvez ouvrir le même numéro d'unité logique physique et charger l'instance de magasin de combinaisons clé-valeur précédente dans l'état où elle était lorsque vous l'avez arrêtée. Pour configurer une instance de magasin de combinaisons clé-valeur en vue de sa persistance lorsqu'elle est arrêtée (ark_delete), définissez le bit `ARK_KV_PERSIST_STORE` dans le paramètre **flags**. Par défaut, une instance de magasin de combinaisons clé-valeur n'est pas configurée en vue de sa persistance. Pour charger l'instance de magasin de combinaisons clé-valeur conservée qui est stockée sur le numéro d'unité logique physique, définissez le bit `ARK_KV_PERSIST_LOAD` dans le paramètre **flags**. Par défaut, l'instance conservée, si elle existe, n'est pas chargée et est remplacée par toute nouvelle donnée conservée.

Seuls les magasins de combinaisons clé-valeur qui sont stockés sur des numéros d'unité logique physiques peuvent être conservés.

Si l'opération aboutit, le paramètre **handle** représente l'instance de magasin de combinaisons clé-valeur nouvellement créée qui est utilisée pour les appels API ultérieurs.

Paramètres

path
Spécifie la carte CAPI, le fichier ou la mémoire pour le magasin de combinaisons clé-valeur.

ark
Spécifie le descripteur représentant le magasin de combinaisons clé-valeur.

flags
Ensemble des indicateurs de bit suivants déterminant les propriétés du magasin de combinaisons clé-valeur :

ARK_KV_VIRTUAL_LUN
Spécifie que le magasin de combinaisons clé-valeur doit utiliser un numéro d'unité logique virtuel créé à partir d'un numéro d'unité logique physique représenté par le fichier spécial.

ARK_KV_PERSIST_STORE
Configure l'instance de magasin de combinaisons clé-valeur qui doit être conservée en cas d'arrêt. Vous pouvez arrêter ou supprimer une instance de magasin de combinaisons clé-valeur avec l'API ark_delete.

ARK_KV_PERSIST_LOAD
Charge la configuration stockée si les données de persistance se trouvent sur le numéro d'unité logique physique.

Valeurs de retour

0 Indique que l'exécution a réussi. Le paramètre **handle** désigne l'instance de magasin de combinaisons clé-valeur nouvellement créée.

EINVAL
Valeur non valide pour l'un des paramètres.

ENOSPC
Mémoire ou stockage flash insuffisant.

ENOTREADY
Le système n'est pas prêt pour la configuration du magasin de combinaisons clé-valeur.

API ark_delete
Objectif

Supprime une instance de magasin de combinaisons clé-valeur.

Syntaxe

```
int ark_delete(ark)
ARK *ark;
```

Description

L'API ark_delete supprime une instance de magasin de combinaisons clé-valeur qui est spécifiée par le paramètre **ark** sur le système hôte. En cas de réussite, toutes les ressources de stockage et mémoire associées sont libérées. De plus, si l'instance ARK est configurée en vue de sa persistance, la configuration est conservée pour que l'instance puisse être chargée ultérieurement.

Paramètres

ark
Descripteur représentant l'instance de magasin de combinaisons clé-valeur.

Valeurs de retour

En cas de réussite, l'API ark_delete nettoie et supprime toutes les ressources associées à l'instance de magasin de combinaisons clé-valeur et renvoie 0. En cas d'échec, l'API ark_delete renvoie l'un des codes d'erreur différents de zéro suivants :

0 L'API s'est exécutée sans erreur. Toutes les ressources associées à l'instance de magasin de combinaisons clé-valeur sont supprimées.

EINVAL
Le descripteur de magasin de combinaisons clé-valeur n'est pas valide.

Valeur différente de zéro
Une erreur est survenue et l'API ne s'est pas exécutée sans erreur.

API ark_set, ark_set_async_cb
Objectif

Ecrit une paire clé-valeur.

Syntaxe

```
int ark_set(ark, klen, key, vlen, val, res)
int ark_set_async_cb(ark, klen, key, vlen, val, callback, dt)

ARK * ark;
uint64_t klen;
void * key;
uint64_t vlen;
void * val;
void *(*callback)(int errcode, uint64_t dt, uint64_t res);
uint64_t dt;
```

Description

L'API ark_set stocke la clé et la valeur dans le magasin pour l'instance de magasin de combinaisons clé-valeur représentée par le paramètre **ark**. L'API ark_set_async_cb s'exécute en mode asynchrone, dans lequel elle revient immédiatement au processus appelant, et l'exécution de l'opération est planifiée. Une fois l'opération exécutée, la fonction `callback` est appelée pour notifier le processus appelant de la fin de l'opération.

Pour une instance de magasin de combinaisons clé-valeur, si la clé est présente, la valeur stockée est remplacée par la valeur `val`.

En cas de réussite, la paire clé-valeur est écrite dans le magasin et le nombre d'octets écrits dans le magasin de combinaisons clé-valeur est renvoyé au processus appelant via le paramètre **res**.

Paramètres

ark
Indique un descripteur qui représente la connexion pour l'instance de magasin de combinaisons clé-valeur.

key
Spécifie la clé dans la paire clé-valeur.

klen
Indique la longueur de la clé en octets.

val
Spécifie la valeur dans la paire clé-valeur.

vlen
Indique la longueur de la valeur en octets.

res
Indique le nombre d'octets écrits dans le magasin de combinaisons clé-valeur en cas de réussite de l'opération d'E-S.

callback
Spécifie la fonction à appeler à la fin de l'opération d'E-S.

dt Indique une valeur de 64 bits pour baliser un appel API asynchrone.

Valeurs de retour

En cas de réussite, les API ark_set et ark_set_async_cb écrivent la paire clé-valeur dans le magasin associé à l'instance de magasin de combinaisons clé-valeur et renvoient le nombre d'octets écrits. La valeur de retour de l'API ark_set indique le statut de l'opération. La valeur de retour de l'API ark_set_async_cb indique si l'opération asynchrone a été acceptée ou rejetée. Le statut est stocké dans le paramètre **errcode** lorsque la fonction `callback` est exécutée. En cas d'échec, les API ark_set et ark_set_async_cb renvoient l'un des codes d'erreur différents de zéro suivants :

EINVAL
Paramètres non valides.

ENOSPC
L'espace restant dans le magasin de combinaisons clé-valeur est insuffisant.

API ark_get, ark_get_async_cb
Objectif

Extrait une valeur pour une clé spécifique.

Syntaxe

```
int ark_get(ark, klen, key, vbuflen, vbuf, voff, res)
int ark_get_async_cb(ark, klen, key, vbuflen, vbuf, voff, callback, dt)

ARK * ark;
uint64_t klen;
void * key;
uint64_t vbuflen;
void * vbuf;
uint64_t voff;
void *(*callback)(int errcode, uint64_t dt, uint64_t res);
uint64_t dt;
```

Description

Les API ark_get et ark_get_async_cb recherchent dans le magasin de combinaisons clé-valeur associé au paramètre **ark** un paramètre **key** spécifique. Si la clé est trouvée, sa valeur est renvoyée dans le paramètre **vbuf**, et un nombre maximal de **vbuflen** octets est écrit dans le magasin de combinaisons clé-valeur, à partir du paramètre de décalage **voff** dans la valeur de la clé. L'API ark_get_async_cb s'exécute en mode asynchrone, dans lequel elle revient immédiatement au processus appelant, et l'exécution de l'opération d'extraction est planifiée. Une fois l'opération terminée, la fonction `callback` est appelée pour notifier le processus appelant de la fin de l'opération.

Si l'API s'exécute sans erreur, la longueur de la valeur de clé est stockée dans le paramètre **res** de la fonction `callback`.

Paramètres

ark
Indique un descripteur qui représente la connexion pour l'instance de magasin de combinaisons clé-valeur.

key
Spécifie la clé dans la paire clé-valeur.

klen
Indique la longueur de la clé en octets.

vbuf
Spécifie le tampon dans lequel stocker la valeur de la clé de la paire clé-valeur.

vbuflen
Spécifie la longueur du tampon **vbuf**.

voff
Spécifie la valeur de décalage dans la clé pour le démarrage de l'opération de lecture.

res
Stocke la taille de la clé en octets si l'API ark_get se termine sans erreur.

callback
Spécifie la fonction `callback` à appeler à la fin de l'opération d'E-S.

dt Spécifie une valeur de 64 bits pour baliser un appel API asynchrone.

Valeurs de retour

En cas de réussite, les API ark_get et ark_get_async_cb renvoient 0. La valeur de retour de l'API ark_get indique le statut de l'opération. La valeur de retour de l'API ark_get_async_cb indique si l'opération asynchrone a été acceptée ou rejetée. Le statut de l'API asynchrone est stocké dans le paramètre **errcode** de la fonction `callback`. En cas d'échec, les API ark_get et ark_get_async_cb renvoient l'un des codes d'erreur différents de zéro suivants :

EINVAL
Paramètres non valides.

ENOENT
Clé introuvable.

ENOSPC
Espace insuffisant dans le tampon mémoire pour stocker la valeur de la clé.

API ark_del, ark_del_async_cb
Objectif

Supprime la valeur associée à une clé spécifique.

Syntaxe

```
int ark_del(ark, klen, key, res)
int ark_del_async_cb(ark, klen, key, callback, dt)

ARK * ark
uint64_t klen;
void * key;
void *(*callback)(int errcode, uint64_t dt, uint64_t res);
uint64_t dt;
```

Description

Les API ark_del et ark_del_async_cb recherchent dans le magasin de combinaisons clé-valeur associé au paramètre **handle** un paramètre **key** spécifique. Si la clé est trouvée, l'API ark_del supprime la valeur du magasin de combinaisons clé-valeur. L'API ark_del_async_cb s'exécute en mode asynchrone, dans lequel elle revient immédiatement au processus appelant, et l'opération de retrait est planifiée. Une fois l'opération terminée, la fonction `callback` est appelée pour notifier le processus appelant de la fin de l'opération.

Si l'API s'exécute sans erreur, la longueur de la valeur de clé est renvoyée au processus appelant dans le paramètre **res** de la fonction `callback`.

Paramètres

ark
Indique un descripteur qui représente la connexion pour l'instance de magasin de combinaisons clé-valeur.

key
Spécifie la clé dans une paire clé-valeur.

klen
Indique la longueur de la clé en octets.

res
Stocke la taille de la clé en octets si cette API se termine sans erreur.

callback
Spécifie la fonction `callback` à appeler à la fin de l'opération d'E-S.

dt Spécifie une valeur de 64 bits pour baliser un appel API asynchrone.

Valeurs de retour

En cas de réussite, les API ark_del et ark_del_async_cb renvoient la valeur 0. La valeur de retour de l'API ark_del indique le statut de l'opération. La valeur de retour de l'API ark_del_async_cb indique si l'opération asynchrone a été acceptée ou rejetée. Le statut de l'API asynchrone est stocké dans le paramètre **errcode** de la fonction `callback`. En cas d'échec, les API ark_del et ark_del_async_cb renvoient l'un des codes d'erreur différents de zéro suivants :

EINVAL
Paramètres non valides.

ENOENT
Clé introuvable.

API ark_exists, ark_exists_async_cb
Objectif

Interroge le magasin de combinaisons clé-valeur afin de vérifier l'existence d'une clé spécifique.

Syntaxe

```
int ark_exist(ark, klen, key, res)
int ark_exist_async_cb(ark, klen, key, callback, dt)

ARK * ark
uint64_t klen;
void * key;
void *(*callback)(int errcode, uint64_t dt, uint64_t res);
uint64_t dt;
```

Description

Les API ark_exists et ark_exists_async_cb recherchent dans le magasin de combinaisons clé-valeur associé au paramètre **ark** un paramètre **key** spécifique. Si la clé est trouvée, l'API ark_exist renvoie la taille de la valeur en octets dans le paramètre **res**. La clé et sa valeur ne sont pas modifiées. L'API ark_exists_async_cb s'exécute en mode asynchrone, dans lequel elle revient immédiatement au processus appelant, et l'opération d'interrogation est planifiée. Une fois l'opération exécutée, la fonction `callback` est appelée pour notifier le processus appelant de la fin de l'opération.

Paramètres

ark
Indique un descripteur qui représente la connexion pour l'instance de magasin de combinaisons clé-valeur.

key
Spécifie la clé dans une paire clé-valeur.

klen
Indique la longueur de la clé en octets.

res
Stocke la taille de la clé en octets si l'API se termine sans erreur.

callback
Spécifie la fonction `callback` à appeler à la fin de l'opération d'E-S.

dt Spécifie une valeur de 64 bits pour baliser un appel API asynchrone.

Valeurs de retour

En cas de réussite, les API ark_exists et ark_exists_async_cb renvoient la valeur 0. La valeur de retour de l'API ark_exists indique le statut de l'opération. La valeur de retour de l'API ark_exists_async_cb indique si l'opération asynchrone a été acceptée ou rejetée. Le statut de l'API asynchrone est stocké dans le paramètre **errcode** de la fonction `callback`. En cas d'échec, les API ark_exists et ark_exists_async_cb renvoient l'un des codes d'erreur différents de zéro suivants :

EINVAL
Paramètres non valides.

ENOENT
Clé introuvable.

API ark_first
Objectif

Renvoie la première clé qui est trouvée dans le magasin de combinaisons clé-valeur ainsi que le descripteur à itérer dans le magasin de combinaisons clé-valeur.

Syntaxe

```
ARI*ark_first(ark, kbuflen, klen, kbuf)
ARK * ark
uint64_t buflen;
int64_t *klen;
void * kbuf;
```

Description

L'API ark_first renvoie la première clé trouvée dans le magasin de combinaisons clé-valeur dans le tampon **kbuf**, ainsi que la taille de clé dans le paramètre **klen**, lorsque la taille de la clé (**klen**) est inférieure à la taille **kbuf** (**kbuflen**).

Si cette API s'exécute sans erreur, un descripteur d'itérateur est renvoyé au processus appelant qui doit être utilisé pour extraire la clé suivante du magasin de combinaisons clé-valeur en appelant l'API ark_next.

Paramètres

ark
Indique un descripteur qui représente la connexion pour l'instance de magasin de combinaisons clé-valeur.

kbuflen
Indique la longueur du paramètre **kbuf**.

klen
Spécifie la taille de la clé renvoyée dans le paramètre **kbuf**.

kbuf
Spécifie le tampon devant contenir la clé.

Valeurs de retour

En cas de réussite, l'API ark_first renvoie un descripteur qui doit être utilisé pour l'itération dans le magasin de combinaisons clé-valeur pour les appels ultérieurs avec l'API ark_next. En cas d'échec, l'API ark_first renvoie la valeur NULL avec l'un des numéros d'erreur suivants :

EINVAL
Paramètres non valides.

ENOSPC
Le paramètre **kbuf** ne dispose pas de suffisamment d'espace pour stocker la clé.

API ark_next
Objectif

Renvoie la clé suivante trouvée dans le magasin de combinaisons clé-valeur.

Syntaxe

```
int ark_next(iter, kbuflen, klen, kbuf)
ARK * iter
uint64_t buflen;
int64_t *klen;
void *kbuf;
```

Description

L'API ark_next renvoie la clé suivante trouvée dans le magasin de combinaisons clé-valeur en fonction du descripteur d'itérateur **iter** dans le tampon **kbuf**, ainsi que la taille de clé dans le paramètre **klen**, lorsque la taille de la clé (**klen**) est inférieure à la taille **kbuf** (**kbuflen**).

En cas de réussite, un descripteur est renvoyé au processus appelant qui doit être utilisé pour extraire la clé suivante du magasin de combinaisons clé-valeur en appelant l'API ark_next. Si la fin du magasin de combinaisons clé-valeur est atteinte, le code d'erreur ENOENT est renvoyé.

Remarque : En raison de la nature dynamique du magasin, certaines des clés qui sont écrites peuvent ne pas être renvoyées.

Paramètres

iter
Spécifie le descripteur d'itérateur à partir duquel lancer la recherche dans le magasin de combinaisons clé-valeur.

kbuf
Spécifie le tampon devant contenir la clé.

kbuflen
Indique la longueur du paramètre **kbuf**.

klen
Spécifie la taille de la clé renvoyée dans le paramètre **kbuf**.

Valeurs de retour

En cas de réussite, l'API ark_next renvoie un descripteur qui doit être utilisé pour l'itération dans le magasin de combinaisons clé-valeur pour les appels ultérieurs avec l'API ark_next. En cas d'échec, l'API ark_next renvoie l'une des valeurs suivantes :

EINVAL
Paramètre non valide.

ENOENT
Fin du magasin atteinte.

API ark_allocated
Objectif

Renvoie le nombre d'octets alloués au magasin.

Syntaxe

```
int ark_allocated(ark, size)
ARK * ark;
uint64_t *size;
```

Description

L'API ark_allocated renvoie le nombre d'octets alloués au magasin de combinaisons clé-valeur via le paramètre **size**.

Paramètres

ark
Spécifie le descripteur représentant le magasin de combinaisons clé-valeur.

size
Contient la taille des blocs alloués au magasin de combinaisons clé-valeur en octets.

Valeurs de retour

0 Indique que l'exécution a réussi.

EINVAL
Indique l'échec en raison d'un paramètre non valide.

API ark_inuse
Objectif

Renvoie le nombre d'octets utilisés dans le magasin de combinaisons clé-valeur.

Syntaxe

```
int ark_inuse(ark, size)
ARK * ark;
uint64_t *size;
```

Description

L'API ark_inuse renvoie le nombre d'octets utilisés dans le magasin de combinaisons clé-valeur via le paramètre **size**.

Paramètres

ark
Spécifie le descripteur représentant le magasin de combinaisons clé-valeur.

size
Contient la taille du magasin utilisé en octets.

Valeurs de retour

0 Indique que l'exécution a réussi.

EINVAL
Indique l'échec en raison d'un paramètre non valide.

API ark_actual
Objectif

Renvoie le nombre d'octets utilisés dans le magasin de combinaisons clé-valeur.

Syntaxe

```
int ark_actual(ark, size)
ARK * ark;
uint64 * size;
```

Description

L'API ark_actual renvoie le nombre d'octets utilisés dans le magasin de combinaisons clé-valeur via le paramètre **size**. Cette API diffère de l'API ark_inuse car elle utilise la taille réelle des clés individuelles et de leurs valeurs au lieu d'allocations de blocs génériques pour stocker ces valeurs.

Paramètres

ark
Spécifie le descripteur représentant le magasin de combinaisons clé-valeur.

size
Contient la taille des blocs utilisés en octets.

Valeurs de retour

0 Indique que l'exécution a réussi. Le paramètre **handle** désigne l'instance de magasin de combinaisons clé-valeur nouvellement créée.

EINVAL
Indique l'échec en raison d'un paramètre non valide.

API ark_fork, ark_fork_done
Objectif

Dévie un magasin de combinaisons clé-valeur pour archivage. Ce service fonctionne pour la plateforme Linux seulement.

Syntaxe

```
int ark_fork(ark)
int ark_fork_done(ark)
ARK * handle;
```

Description

Les API ark_fork et ark_fork_done sont appelées par le processus de magasin de combinaisons clé-valeur parent pour préparer le magasin de combinaisons clé-valeur en vue de sa déviation (division en plusieurs processus), pour dévier le processus enfant, et pour nettoyer l'état de l'appel une fois le processus enfant arrêté. L'API ark_fork dévie un processus enfant et une fois son exécution terminée, renvoie l'ID de processus du processus enfant au processus parent et renvoie 0 au processus enfant. Une fois que le processus parent a détecté que le processus enfant est arrêté, l'API ark_fork_done est appelée pour nettoyer tout état de l'appel ark_fork.

Remarque : l'API ark_fork échoue si des commandes asynchrones existent. Le service fonctionne pour la plateforme Linux seulement.

Paramètres

ark
Spécifie le descripteur représentant le magasin de combinaisons clé-valeur.

Valeurs de retour

0 Indique que l'exécution a réussi.

EINVAL
Indique l'échec en raison d'un paramètre non valide.

EBUSY
Indique l'échec en raison d'opérations asynchrones en suspens.

ENOMEM
Indique l'échec car l'espace est insuffisant pour cloner le magasin.

API ark_random
Objectif

Renvoie une clé aléatoire depuis le magasin de combinaisons clé-valeur.

Syntaxe

```
int ark_random(ark, kbuflen, klen, kbuf)
ARK * ark;
uint64_t kbuflen
int64_t *klen;
void * kbuf;
```

Description

L'API ark_random renvoie une clé aléatoire depuis le magasin de combinaisons clé-valeur en fonction du descripteur **ark** dans le tampon **kbuf**, ainsi que la taille de la clé dans le paramètre **klen**, lorsque la taille de clé (**klen**) est inférieure à la taille **kbuf** (**kbuflen**).

Paramètres

ark
Spécifie le descripteur représentant le magasin de combinaisons clé-valeur.

kbuflen
Contient la taille du magasin de combinaisons clé-valeur en octets.

klen
Spécifie la taille de la clé renvoyée dans le paramètre `kbuf`.

kbuf
Spécifie le tampon devant contenir la clé.

Valeurs de retour

0 Indique que l'exécution a réussi.

EINVAL
Indique l'échec en raison d'un paramètre non valide.

API ark_count
Objectif

Renvoie le nombre de clés qui se trouvent dans le magasin de combinaisons clé-valeur.

Syntaxe

```
int ark_count(ark, count)
ARK * ark;
int * count;
```

Description

L'API ark_count renvoie le nombre total de clés qui se trouvent dans le magasin de combinaisons clé-valeur en fonction du descripteur **ark** et stocke le résultat dans le paramètre **count**.

Paramètres

ark
Spécifie le descripteur représentant le magasin de combinaisons clé-valeur.

count
Spécifie le nombre de clés qui se trouvent dans le magasin de combinaisons clé-valeur.

Valeurs de retour

0 Indique que l'exécution a réussi.

EINVAL
Indique l'échec en raison d'un paramètre non valide.

API ark_stats
Objectif

Renvoie le nombre d'opérations d'E-S de clé-valeur ainsi que le nombre d'opérations d'E-S par bloc.

Syntaxe

```
#include <arkdb.h>

int ark_stats(ARK *ark, uint64_t *ops, uint64_t *ios);
```

Description

L'API ark_stats renvoie le nombre total d'opérations d'E-S de clé-valeur via le paramètre **ops**, ainsi que le nombre total d'opérations d'E-S par bloc via le paramètre **ios**.

Paramètres

ark
Spécifie le descripteur représentant le magasin de combinaisons clé-valeur.

ops
Indique le nombre total d'opérations d'E-S de clé-valeur.

ios
Indique le nombre total d'opérations d'E-S par bloc.

Valeurs de retour

0 Indique que l'exécution a réussi.

EINVAL
Indique qu'une erreur est survenue.

Remarques

Le présent document peut contenir des informations ou des références concernant certains produits, logiciels ou services IBM non annoncés dans ce pays. Consultez votre interlocuteur commercial IBM local pour plus d'informations sur les produits et services disponibles dans votre pays. Toute référence à un produit, logiciel ou service IBM n'implique pas que seul ce produit, logiciel ou service IBM puisse être utilisé. Tout autre élément fonctionnellement équivalent peut être utilisé, s'il n'enfreint aucun droit d'IBM. Il est de la responsabilité de l'utilisateur d'évaluer et de vérifier lui-même les installations et applications réalisées avec des produits, logiciels ou services non expressément référencés par IBM.

IBM peut détenir des brevets ou des demandes de brevet couvrant les produits mentionnés dans le présent document. La remise de ce document ne vous accorde aucun droit de licence sur ces brevets ou demandes de brevet. Si vous désirez recevoir des informations concernant l'acquisition de licences, veuillez en faire la demande par écrit à l'adresse suivante :

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
USA

Pour le Canada, veuillez adresser votre courrier à :

IBM Director of Commercial Relations
IBM Canada Ltd.
3600 Steeles Avenue East
Markham, Ontario
L3R 9Z7
Canada

Les informations sur les licences concernant les produits utilisant un jeu de caractères double octet peuvent être obtenues par écrit à l'adresse suivante :

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japon

LE PRESENT DOCUMENT EST LIVRE "EN L'ETAT" SANS AUCUNE GARANTIE EXPLICITE OU IMPLICITE. IBM DECLINE NOTAMMENT TOUTE RESPONSABILITE RELATIVE A CES INFORMATIONS EN CAS DE CONTREFAÇON AINSI QU'EN CAS DE DEFAUT D'APTITUDE A L'EXECUTION D'UN TRAVAIL DONNE. Certaines juridictions n'autorisent pas l'exclusion des garanties implicites, auquel cas l'exclusion ci-dessus ne vous sera pas applicable.

Le présent document peut contenir des inexactitudes ou des coquilles. Ce document est mis à jour périodiquement. Chaque nouvelle édition inclut les mises à jour. IBM peut, à tout moment et sans préavis, modifier les produits et logiciels décrits dans ce document.

Les références à des sites Web non IBM sont fournies à titre d'information uniquement et n'impliquent en aucun cas une adhésion aux données qu'ils contiennent. Les éléments figurant sur ces sites Web ne font pas partie des éléments du présent produit IBM et l'utilisation de ces sites relève de votre seule responsabilité.

IBM pourra utiliser ou diffuser, de toute manière qu'elle jugera appropriée et sans aucune obligation de sa part, tout ou partie des informations qui lui seront fournies.

Les licenciés souhaitant obtenir des informations permettant : (i) l'échange des données entre des logiciels créés de façon indépendante et d'autres logiciels (dont celui-ci), et (ii) l'utilisation mutuelle des données ainsi échangées, doivent adresser leur demande à :

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
USA

Ces informations peuvent être soumises à des conditions particulières, prévoyant notamment le paiement d'une redevance.

Le logiciel sous licence décrit dans ce document et tous les éléments sous licence disponibles s'y rapportant sont fournis par IBM conformément aux dispositions du Livret contractuel IBM, des Conditions Internationales d'Utilisation de Logiciels IBM ou de tout autre accord équivalent.

Les données de performance et les clients cités sont présentés à titre d'exemple uniquement. Les performances réelles peuvent varier en fonction des configurations et des conditions d'exploitation spécifiques.

Les informations concernant des produits non IBM ont été obtenues auprès des fournisseurs de ces produits, par l'intermédiaire d'annonces publiques ou via d'autres sources disponibles. IBM n'a pas testé ces produits et ne peut confirmer l'exactitude de leurs performances ni leur compatibilité. Elle ne peut recevoir aucune réclamation concernant des produits non IBM. Toute question concernant les performances de produits non IBM doit être adressée aux fournisseurs de ces produits.

Toute instruction relative aux intentions d'IBM pour ses opérations à venir est susceptible d'être modifiée ou annulée sans préavis, et doit être considérée uniquement comme un objectif.

Tous les tarifs indiqués sont les prix de vente actuels suggérés par IBM et sont susceptibles d'être modifiés sans préavis. Les tarifs appliqués peuvent varier selon les revendeurs.

Ces informations sont fournies uniquement à titre de planification. Elles sont susceptibles d'être modifiées avant la mise à disposition des produits décrits.

Le présent document peut contenir des exemples de données et de rapports utilisés couramment dans l'environnement professionnel. Ces exemples mentionnent des noms fictifs de personnes, de sociétés, de marques ou de produits à des fins illustratives ou explicatives uniquement. Toute ressemblance avec des noms de personnes ou de sociétés serait purement fortuite.

LICENCE DE COPYRIGHT :

Le présent logiciel contient des programmes exemples d'application en langage source destinés à illustrer les techniques de programmation sur différentes plates-formes d'exploitation. Vous avez le droit de copier, de modifier et de distribuer ces programmes exemples sous quelque forme que ce soit et sans paiement d'aucune redevance à IBM à des fins de développement, d'utilisation, de vente ou de distribution de programmes d'application conformes à l'interface de programme d'application de la plateforme pour lesquels ils ont été écrits. Ces exemples de programme n'ont pas été rigoureusement testés dans toutes les conditions. Par conséquent, IBM ne peut garantir expressément ou implicitement la fiabilité, la maintenabilité ou le fonctionnement de ces programmes. Les programmes exemples sont livrés "EN L'ETAT", sans garantie d'aucune sorte. IBM ne sera en aucun cas responsable des dommages liés à l'utilisation des programmes exemples.

Toute copie totale ou partielle de ces programmes exemples et des oeuvres qui en sont dérivées doit comprendre une notice de copyright, libellée comme suit :

© (le nom de votre société) (année).

Des segments de ce code sont dérivés des Programmes exemples d'IBM Corp.

© Copyright IBM Corp. _indiquez l'année ou les années_.

Politique de confidentialité

Les logiciels IBM®, y compris les Logiciels sous forme de services ("Offres Logiciels"), peuvent utiliser des cookies ou d'autres technologies pour collecter des informations sur l'utilisation des produits, améliorer l'acquis utilisateur, personnaliser les interactions avec celui-ci, ou dans d'autres buts. Bien souvent, aucune information personnelle identifiable n'est collectée par les Offres Logiciels. Certaines Offres Logiciels vous permettent cependant de le faire. Si la présente Offre Logiciels utilise des cookies pour collecter des informations personnelles identifiables, des informations spécifiques sur cette utilisation sont fournies ci-dessous.

La présente Offre Logiciels n'utilise pas de cookies, ni d'autres technologies, pour collecter des informations personnelles identifiables.

Si les configurations déployées de cette Offre Logiciels vous permettent, en tant que client, de collecter des informations permettant d'identifier les utilisateurs par l'intermédiaire de cookies ou par d'autres techniques, vous devez solliciter un avis juridique sur la réglementation applicable à ce type de collecte, notamment en termes d'information et de consentement.

Pour plus d'informations sur l'utilisation à ces fins des différentes technologies, y compris celle des cookies, consultez les Points principaux de la Déclaration IBM de confidentialité sur Internet à l'adresse http://www.ibm.com/privacy, la section "Cookies, pixels espions et autres technologies" de la Déclaration IBM de confidentialité sur Internet à l'adresse http://www.ibm.com/privacy/details, ainsi que la page "IBM Software Products and Software-as-a-Service Privacy Statement" à l'adresse http://www.ibm.com/software/info/product-privacy.

Marques

IBM, le logo IBM et ibm.com sont des marques d'International Business Machines Corp., dans de nombreux pays. Les autres noms de produits et de services peuvent être des marques d'IBM ou d'autres sociétés. La liste actualisée de toutes les marques d'IBM est disponible sur la page Web Copyright and trademark information à www.ibm.com/legal/copytrade.shtml.

Linux est une marque de Linus Torvalds aux Etats-Unis et/ou dans certains autres pays.

Index

A

C