NOM

fork − Créer un processus fils

SYNOPSIS

#include <unistd.h>

pid_t fork(void);

DESCRIPTION

fork() crée un nouveau processus en copiant le processus appelant. Le nouveau processus, qu'on appelle *fils* (« child »), est une copie exacte du processus appelant, qu'on appelle *père* ou *parent*, avec les exceptions suivantes :

* Le fils a son propre identifiant de processus unique, et ce PID ne correspond à l'identifiant d'aucun groupe de processus existant (**setpgid**(2)).
* L'identifiant de processus parent (PPID) du fils est l'identifiant de processus (PID) du père.
* Le fils n'hérite pas des verrouillages mémoire du père (**mlock**(2), **mlockall**(2)).
* Les utilisations de ressources (**getrusage**(2)) et les compteurs de temps processeur (**times**(2)) sont remis à zéro dans le fils.
* L'ensemble de signaux en attente dans le fils est initialement vide (**sigpending**(2)).
* Le fils n'hérite pas des opérations sur les sémaphores de son père (**semop**(2)).
* Le fils n'hérite pas des verrous d'enregistrements de son père (**fcntl**(2)).
* Le fils n'hérite pas des temporisations de son père (**setitimer**(2), **alarm**(2), **timer_create**(2)).
* Le fils n'hérite pas des opérations d'E/S asynchrones en cours de son père (**aio_read**(3), **aio_write**(3)) et n'hérite d'aucun contexte d'E/S asynchrone de son père (consultez **io_setup**(2)).

Les attributs du processus listés ci-dessus sont tous spécifiés dans POSIX.1−2001. Les processus parent et fils diffèrent également par les propriétés spécifiques Linux suivantes :

* Le fils n'hérite pas des notifications de modification de répertoire (dnotify) de son père (voir la description de **F_NOTIFY** dans **fcntl**(2)).
* Le drapeau **PR_SET_PDEATHSIG** de **prctl**(2) est réinitialisé, de manière à ce que le fils ne reçoive pas de signal lorsque son père se termine.
* La valeur de temporisation relâchée par défaut est définie à la valeur de temporisation relâchée actuelle de son père. Veuillez consulter la description de **PR_SET_TIMERSLACK** dans **prctl**(2).
* Les projections en mémoire qui ont été marquées avec l'attribut **MADV_DONTFORK** de **madvise**(2) ne sont pas hérités lors d'un **fork**().
* Le signal de terminaison du fils est toujours **SIGCHLD** (consultez **clone**(2)).
* Les bits de permission d'accès au port indiqués par **ioperm**(2) ne sont pas hérités par le fils ; le fils doit activer avec **ioperm**(2) les bits dont il a besoin.

Notez également les points suivants :

* Le processus fils est créé avec un unique thread — celui qui a appelé **fork**(). L'espace d'adressage virtuel complet du parent est copié dans le fils, y compris l'état des mutex, variables de condition, et autres objets de pthreads ; l'utilisation de **pthread_atfork**(3) peut être utile pour traiter les problèmes que cela peut occasionner.
* Le fils hérite de copies des descripteurs de fichier ouverts du père. Chaque descripteur de fichier du fils renvoie à la même description de fichier ouvert (consultez **open**(2)) que le descripteur de fichier correspondant dans le processus parent. Cela signifie que les deux descripteurs partagent les attributs d'état du fichier, le décalage, et les attributs d'E/S liés aux signaux (voir la description de **F_SETOWN** et **F_SETSIG** dans **fcntl**(2)).

* Le fils hérite de copies des descripteurs files de messages ouvertes dans le père (consultez **mq_overview**(7)). Chaque descripteur dans le fils renvoie à la même description de file de messages ouverte que le descripteur correspondant dans le père. Cela signifie que les deux descripteurs partagent leurs attributs (*mq_flags*).
* Le fils hérite d'une copie de l'ensemble des flux de répertoire ouverts par le parent (consultez **opendir**(3)). POSIX.1−2001 indique que les flux de répertoire correspondant dans le parent ou l'enfant *peuvent* partager le positionnement du flux de répertoire ; sous Linux/glibc, ce n'est pas le cas.

VALEUR RENVOYÉE

En cas de succès, le PID du fils est renvoyé au parent, et 0 est renvoyé au fils. En cas d'échec −1 est renvoyé au parent, aucun processus fils n'est créé, et *errno* contient le code d'erreur.

ERREURS

EAGAIN
fork() ne peut pas allouer assez de mémoire pour copier la table des pages du père et allouer une structure de tâche pour le fils.

EAGAIN
Il n'a pas été possible de créer un nouveau processus car la limite ressource **RLIMIT_NPROC** de l'appelant a été rencontrée. Pour franchir cette limite, le processus doit avoir au moins l'une des deux capacités **CAP_SYS_ADMIN** ou **CAP_SYS_RESOURCE**.

ENOMEM
fork() a échoué car le noyau n'a plus assez de mémoire.

ENOSYS
fork() n'est pas supporté sur cette plate−forme (par exemple sur du matériel sans unité de gestion mémoire).

CONFORMITÉ

SVr4, BSD 4.3, POSIX.1−2001.

NOTES

Sous Linux, **fork**() est implémenté en utilisant une méthode de copie à l'écriture. Ceci consiste à ne faire la véritable duplication d'une page mémoire que lorsqu'un processus en modifie une instance. Tant qu'aucun des deux processus n'écrit dans une page donnée, celle-ci n'est pas vraiment dupliquée. Ainsi les seules pénalisations induites par fork sont le temps et la mémoire nécessaires à la copie de la table des pages du parent ainsi que la création d'une structure de tâche pour le fils.

Depuis la version 2.3.3, plutôt que d'invoquer l'appel système **fork**() du noyau, l'enveloppe **fork**() de la glibc qui est fournie comme faisant partie de l'implémentation de threading NPTL invoque **clone**(2) avec des attributs qui fournissent le même effet que l'appel système traditionnel (un appel à **fork**() est équivalent à un appel à **clone**(2) avec *flags* valant exactement **SIGCHLD**). L'enveloppe de la glibc invoque tous les gestionnaires de bifurcation (« fork ») établis avec **pthread_atfork**(3).

EXEMPLE

Consultez **pipe**(2) et **wait**(2).

VOIR AUSSI

clone(2), **execve**(2), **exit**(2), **setrlimit**(2), **unshare**(2), **vfork**(2), **wait**(2), **daemon**(3), **capabilities**(7), **credentials**(7)

COLOPHON

Cette page fait partie de la publication 3.65 du projet *man−pages* Linux. Une description du projet et des instructions pour signaler des anomalies peuvent être trouvées à l'adresse http://www.kernel.org/doc/man−pages/.

TRADUCTION

Depuis 2010, cette traduction est maintenue à l'aide de l'outil po4a <http://po4a.alioth.debian.org/> par l'équipe de traduction francophone au sein du projet perkamon <http://perkamon.alioth.debian.org/>.

Christophe Blaess <http://www.blaess.fr/christophe/> (1996-2003), Alain Portal <http://manpagesfr.free.fr/> (2003-2006). Julien Cristau et l’équipe francophone de traduction de Debian (2006-2009).

Veuillez signaler toute erreur de traduction en écrivant à <debian−l10n−french@lists.debian.org> ou par un rapport de bogue sur le paquet **manpages−fr**.

Vous pouvez toujours avoir accès à la version anglaise de ce document en utilisant la commande « **man −L C** <*section*> <*page_de_man*> ».