Definition des fonctions autorises

free, fork, sigaction, getenv, tgetent, tgetflag, tgetnum et tgetstr

Voici une description de chaque fonction :

- `readline` est une fonction de la bibliothèque `readline` qui permet de lire une ligne de texte à partir de l'entrée standard. Elle peut être utilisée pour implémenter une interface de ligne de commande interactive.
- `rl_clear_history` est une fonction de la bibliothèque `readline` qui permet de effacer l'historique des lignes de commande entrées par l'utilisateur.
- `rl_on_new_line` est une fonction de la bibliothèque `readline` qui est appelée chaque fois qu'une nouvelle ligne de commande est lue à partir de l'entrée standard. Elle peut être utilisée pour mettre à jour l'interface de ligne de commande en fonction de la nouvelle entrée de l'utilisateur.
- `rl_replace_line` est une fonction de la bibliothèque `readline` qui permet de remplacer la ligne de commande actuellement affichée par une nouvelle ligne de commande.
- `rl_redisplay` est une fonction de la bibliothèque `readline` qui permet de mettre à jour l'affichage de l'interface de ligne de commande en fonction des modifications apportées à la ligne de commande actuelle.
- `add_history` est une fonction de la bibliothèque `readline` qui permet d'ajouter une ligne de commande à l'historique des lignes de commande entrées par l'utilisateur.
- `printf` est une fonction de la bibliothèque `stdio.h` qui permet d'afficher du texte à l'écran. Elle peut être utilisée pour afficher des messages de debug ou des résultats de calcul.
- `malloc` est une fonction de la bibliothèque `stdlib.h` qui permet de réserver de l'espace mémoire dynamiquement. Elle prend en argument la taille de l'espace mémoire à réserver et renvoie un pointeur vers l'adresse de début de cet espace mémoire.

- `write` est une fonction de la bibliothèque `unistd.h` qui permet d'écrire des données dans un descripteur de fichier ouvert. Elle prend en argument le descripteur de fichier, un pointeur vers les données à écrire et la taille des données à écrire.
- `access` est une fonction de la bibliothèque `unistd.h` qui permet de vérifier les permissions d'accès à un fichier ou un dossier. Elle prend en argument le nom du fichier ou du dossier et un masque de permissions, et renvoie 0 si l'accès est accordé ou un code d'erreur sinon.
- `open` est une fonction de la bibliothèque `unistd.h` qui permet d'ouvrir un fichier ou un dossier. Elle prend en argument le nom du fichier ou du dossier et un masque de permissions, et renvoie un descripteur de fichier si l'ouverture a réussi ou -1 en cas d'échec.
- `read` est une fonction de la bibliothèque `unistd.h` qui permet de lire des données à partir d'un descripteur de fichier ouvert. Elle prend en argument le descripteur de fichier, un pointeur vers un tampon où les données lues seront stockées et la taille du tampon, et renvoie le nombre de bytes lus.
- `close` est une fonction de la bibliothèque `unistd.h` qui permet de fermer un descripteur de fichier ouvert. Elle prend en argument le descripteur de fichier et renvoie 0 si la fermeture a réussi ou -1 en cas d'échec.
- `wait` est une fonction de la bibliothèque `sys/wait.h` qui permet au processus appelant d'attendre la terminaison d'un processus fils. Elle renvoie le statut de terminaison du processus fils ou -1 en cas d'erreur.
- `waitpid` est une fonction de la bibliothèque `sys/wait.h` qui permet au processus appelant d'attendre la terminaison d'un processus fils spécifique. Elle prend en argument le PID du processus fils à attendre et un pointeur vers un tampon où le statut de terminaison sera stocké, et renvoie le PID du processus fils terminé ou -1 en cas d'erreur.
- `wait3` et `wait4` sont des fonctions de la bibliothèque `sys/wait.h` qui permettent au processus appelant d'attendre la terminaison d'un processus fils et de récupérer des informations supplémentaires sur le processus fils terminé. Elles prennent en argument un pointeur vers un tampon où le statut de terminaison et les informations supplémentaires seront stockées, et renvoient le PID du processus fils terminé ou -1 en cas d'erreur.

- `signal` est une fonction de la bibliothèque `signal.h` qui permet de définir le comportement d'un processus lorsqu'il reçoit un signal. Elle prend en argument le numéro du signal et une fonction de gestionnaire de signal, et renvoie le gestionnaire de signal précédemment défini pour le signal spécifié.
- `sigemptyset` est une fonction de la bibliothèque `signal.h` qui permet de créer un ensemble de signaux vide. Elle prend en argument un pointeur vers un tampon d'ensemble de signaux et renvoie 0 en cas de succès ou -1 en cas d'erreur.
- `sigaddset` est une fonction de la bibliothèque `signal.h` qui permet d'ajouter un signal à un ensemble de signaux. Elle prend en argument un pointeur vers un tampon d'ensemble de signaux et le numéro du signal à ajouter, et renvoie 0 en cas de succès ou -1 en cas d'erreur.
- `kill` est une fonction de la bibliothèque `signal.h` qui permet d'envoyer un signal à un processus. Elle prend en argument le PID du processus cible et le numéro du signal à envoyer, et renvoie 0 en cas de succès ou -1 en cas d'erreur.
- `exit` est une fonction de la bibliothèque `stdlib.h` qui permet de quitter un programme en renvoyant un code de statut. Elle prend en argument le code de statut à renvoyer et ne renvoie jamais de valeur.
- `getcwd` est une fonction de la bibliothèque `unistd.h` qui permet de récupérer le chemin du dossier courant. Elle prend en argument un pointeur vers un tampon où le chemin sera stocké et la taille du tampon, et renvoie le pointeur vers le tampon si la récupération a réussi ou NULL en cas d'erreur.
- `chdir` est une fonction de la bibliothèque `unistd.h` qui permet de changer de dossier courant. Elle prend en argument le chemin du dossier cible et renvoie 0 en cas de succès ou -1 en cas d'erreur.
- `stat`, `lstat` et `fstat` sont des fonctions de la bibliothèque `sys/stat.h` qui permettent de récupérer des informations sur un fichier ou un dossier. Elles prennent en argument le nom du fichier ou du dossier et un pointeur vers un tampon où les informations seront stockées, et renvoient 0 en cas de succès ou -1 en cas d'erreur. `stat` récupère les informations sur le fichier ou le dossier réel, tandis que `lstat` récupère les informations sur le lien symbolique lui-même et `fstat` récupère les informations sur un descripteur de fichier ouvert.

- `unlink` est une fonction de la bibliothèque `unistd.h` qui permet de supprimer un fichier. Elle prend en argument le nom du fichier à supprimer et renvoie 0 en cas de succès ou -1 en cas d'erreur.
- `execve` est une fonction de la bibliothèque `unistd.h` qui permet de remplacer le code exécutable d'un processus par un nouveau programme. Elle prend en argument le nom du programme à exécuter, un tableau de arguments et un tableau d'environnement, et ne renvoie jamais de valeur.
- `dup` et `dup2` sont des fonctions de la bibliothèque `unistd.h` qui permettent de dupliquer un descripteur de fichier. `dup` prend en argument un descripteur de fichier et renvoie le nouveau descripteur de fichier dupliqué, tandis que `dup2` prend en argument un descripteur de fichier source et un descripteur de fichier cible et renvoie le descripteur de fichier cible dupliqué. Si le descripteur de fichier cible est déjà ouvert, il est fermé avant d'être dupliqué.
- `pipe` est une fonction de la bibliothèque `unistd.h` qui permet de créer un tube de communication entre deux processus. Elle prend en argument un tableau de deux entiers et remplit le premier avec le descripteur de fichier d'écriture et le second avec le descripteur de fichier de lecture. Elle renvoie 0 en cas de succès ou -1 en cas d'erreur.
- `opendir` est une fonction de la bibliothèque `dirent.h` qui permet d'ouvrir un dossier. Elle prend en argument le nom du dossier à ouvrir et renvoie un pointeur vers un tampon de type `DIR` si l'ouverture a réussi ou NULL en cas d'échec.
- `readdir` est une fonction de la bibliothèque `dirent.h` qui permet de lire le contenu d'un dossier ouvert. Elle prend en argument un pointeur vers un tampon de type `DIR` et renvoie un pointeur vers un tampon de type `dirent` qui contient les informations sur l'élément lu ou NULL si la lecture est terminée.
- `closedir` est une fonction de la bibliothèque `dirent.h` qui permet de fermer un dossier ouvert. Elle prend en argument un pointeur vers un tampon de type `DIR` et renvoie 0 en cas de succès ou -1 en cas d'erreur.
- `strerror` est une fonction de la bibliothèque `string.h` qui permet de récupérer le message d'erreur correspondant à un code d'erreur.
- `perror` est une fonction de la bibliothèque `stdio.h` qui permet d'afficher un message d'erreur sur la sortie d'erreur standard, précédé de la chaîne de

caractères spécifiée en argument. Elle ne renvoie aucune valeur.

- `isatty` est une fonction de la bibliothèque `unistd.h` qui permet de vérifier si un descripteur de fichier correspond à un terminal. Elle prend en argument un descripteur de fichier et renvoie 1 si le descripteur de fichier correspond à un terminal, 0 sinon.
- `ttyname` est une fonction de la bibliothèque `unistd.h` qui permet de récupérer le nom du terminal associé à un descripteur de fichier. Elle prend en argument un descripteur de fichier et renvoie un pointeur vers une chaîne de caractères qui contient le nom du terminal si le descripteur de fichier correspond à un terminal, NULL sinon.
- `ttyslot` est une fonction de la bibliothèque `unistd.h` qui permet de récupérer l'index du terminal dans la table des terminaux. Elle prend en argument un descripteur de fichier et renvoie l'index du terminal si le descripteur de fichier correspond à un terminal, -1 sinon.
- `ioctl` est une fonction de la bibliothèque `sys/ioctl.h` qui permet d'envoyer des commandes d'E/S génériques à un descripteur de fichier. Elle prend en argument un descripteur de fichier, une commande d'E/S et un pointeur vers un tampon où les données seront stockées, et renvoie 0 en cas de succès ou -1 en cas d'erreur.
- `tgoto` est une fonction de la bibliothèque `cursed.h` qui permet de générer une chaîne de caractères de contrôle de terminal qui déplace le curseur à une position spécifique. Elle prend en argument une chaîne de caractères qui spécifie la manière de déplacer le curseur et deux entiers qui indiquent la ligne et la colonne cibles, et renvoie une chaîne de caractères qui contient la chaîne de contrôle générée ou NULL si le terminal ne supporte pas le déplacement du curseur.
- `tputs` est une fonction de la bibliothèque `cursed.h` qui permet d'envoyer une chaîne de contrôle de terminal à un terminal. Elle prend en argument une chaîne de caractères qui contient la chaîne de contrôle et un entier qui indique le nombre de lignes à afficher, et renvoie l'entier 0 si l'envoi a réussi ou EOF en cas d'échec.
- `free` est une fonction de la bibliothèque `stdlib.h` qui permet de libérer la mémoire allouée dynamiquement par la fonction `malloc`. Elle prend en argument un pointeur vers un bloc de mémoire alloué dynamiquement et ne renvoie aucune valeur. Cette fonction doit être appelée pour libérer la mémoire allouée dynamiquement une fois qu'elle n'est plus nécessaire, afin d'éviter les fuites de mémoire.

- `fork` est une fonction de la bibliothèque `unistd.h` qui permet de créer un nouveau processus en copiant l'état du processus appelant. Elle ne prend aucun argument et renvoie 0 au processus enfant et le PID du processus enfant au processus parent. Cette fonction est souvent utilisée pour créer des processus qui s'exécutent en parallèle ou pour créer des processus qui s'exécutent en arrière-plan.
- `sigaction` est une fonction de la bibliothèque `signal.h` qui permet de configurer le traitement d'un signal. Elle prend en argument le numéro du signal à configurer, un pointeur vers une structure de type `struct sigaction` qui contient les nouvelles actions à effectuer lorsque le signal est reçu, et un pointeur vers une structure de type `struct sigaction` qui sera rempli avec les actions précédemment configurées pour le signal. Cette fonction renvoie 0 en cas de succès ou -1 en cas d'erreur.
- `getenv` est une fonction de la bibliothèque `stdlib.h` qui permet de récupérer la valeur d'une variable d'environnement. Elle prend en argument une chaîne de caractères qui spécifie le nom de la variable et renvoie une chaîne de caractères qui contient la valeur de la variable ou NULL si la variable n'existe pas.
- `tgetent` est une fonction de la bibliothèque `cursed.h` qui permet de récupérer les informations générales sur un terminal. Elle prend en argument un tampon de type `char` qui doit être assez grand pour contenir les informations sur le terminal et une chaîne de caractères qui contient le nom du terminal, et renvoie le nombre de caractères lus si la récupération a réussi ou -1 en cas d'échec.
- `tgetflag` est une fonction de la bibliothèque `cursed.h` qui permet de récupérer une information booléenne sur un terminal. Elle prend en argument une chaîne de caractères qui spécifie l'information à récupérer et renvoie un entier qui indique si l'information est présente ou non. Si l'information est présente, la fonction renvoie 1, sinon elle renvoie 0.
- `tgetnum` est une fonction de la bibliothèque `cursed.h` qui permet de récupérer une information numérique sur un terminal. Elle prend en argument une chaîne de caractères qui spécifie l'information à récupérer et renvoie un entier qui contient la valeur de l'information ou -1 si l'information n'est pas présente.
- `tgetstr`est une fonction de la bibliothèque `cursed.h` qui permet de récupérer une information de chaîne de caractères sur un terminal. Elle prend en argument une chaîne de caractères qui spécifie l'information à récupérer et renvoie une chaîne de

caractères qui contient la valeur de l'information ou NULL si l'information n'est pas présente.

Pour utiliser ces fonctions, il faut d'abord appeler `tgetent` avec un tampon assez grand pour contenir les informations sur le terminal et le nom du terminal, puis appeler `tgetflag`, `tgetnum` ou `tgetstr` avec les chaînes de caractères qui spécifient l'information à récupérer. Si l'appel à `tgetent` a réussi, les autres fonctions renverront les informations demandées si elles sont présentes, sinon elles renverront une valeur spécifique indiquant l'absence d'information (NULL pour `tgetstr`, 0 pour `tgetflag` et -1 pour `tgetnum`). Il est recommandé de consulter la documentation de la bibliothèque `cursed.h` pour obtenir la liste des chaînes de caractères valides pour chaque fonction.