Getenv : char *getenv(const char *_name_) : recherche dans la liste des variables d'environnement un variable nommée _name_, et renvoie un pointeur sur la chaîne _value_ correspondante.

Readline Library

Fonction	Prototype	Résumé
readline	char *readline *(char *prompt)*	Affiche un prompt et return la ligne lu par la fonction, la ligne retournée est malloc est devra être free
rl_clear_history	void **rl_clear_history** *(void)*	Clear the history list by deleting all of the entries, it frees private data Readline saves in the history list.
rl_on_new_line	int **rl_on_new_line** *(void)*	Tell the update functions that we have moved onto a new (empty) line, usually after outputting a newline.
rl_redisplay	void **rl_redisplay** *(void)***pid_t fork(void)**	Change what's displayed on the screen to reflect the current contents of `rl_line_buffer`.
Add_history	void **add_history** *(char *line)*	Add a new line as the history
rl_replace_line	void **rl_replace_line** *(const char *text, int clear_undo)*	Replace the contents of `rl_line_buffer` with *text*. The point and mark are preserved, if possible. If *clear_undo* is non-zero, the undo list associated with the current line is cleared.

https://man7.org/linux/man-pages/man3/readline.3.html : *man readline*
https://tiswww.case.edu/php/chet/readline/readline.html#IDX220 : *gnu readline library*

Process func

Fork	**pid_t fork(void)**	duplique le processus appellant, le fils est une copie identique, Retourne 0 si l'on se trouve dans le processus enfant, et un chiffre positif si l'on est dans le parent http://manpagesfr.free.fr/man/man2/fork.2.html
Wait	**pid_t wait(int **status*)**	L'appel système **wait**() suspend l'exécution du processus appelant jusqu'à ce que l'un de ses fils se termine http://manpagesfr.free.fr/man/man2/wait.2.html
Waitpid	**pid_t waitpid(pid_t** *pid*, **int *status**, **int** *options*)	L'appel système **waitpid**() suspend l'exécution du processus appelant jusqu'à ce que le fils spécifié par son *pid* ait changé d'état. http://manpagesfr.free.fr/man/man2/wait.2.html
Wait3	**pid_t wait3(int** **status*, **int** *options*, **struct rusage** **rusage*)	Fonctionne comme waitpid, sauf qu'elle attend que n'importe lequel des fils se termine, et renvoie plusieurs information dans la structure **rusage** http://manpagesfr.free.fr/man/man2/wait4.2.html
Wait4	**pid_t wait4(pid_t** *pid*, **int** **status*, **int** *options*, **struct rusage** **rusage*)	Fonctionne comme waitpid, sauf qu'elle renvoie des information dans la structure **rusage** http://manpagesfr.free.fr/man/man2/wait4.2.html

```
struct rusage {
  struct timeval ru_utime; /* Temps utilisateur écoulé         */
  struct timeval ru_stime; /* Temps système écoulé             */
  long  ru_maxrss;         /* Taille résidente maximale        */
  long  ru_ixrss;          /* Taille de mémoire partagée       */
  long  ru_idrss;          /* Taille des données non partagées */
  long  ru_isrss;          /* Taille de pile                   */
  long  ru_minflt;         /* Demandes de pages                */
  long  ru_majflt;         /* Nombre de fautes de pages        */
  long  ru_nswap;          /* Nombre de swaps                  */
  long  ru_inblock;        /* Nombre de lectures de blocs      */
  long  ru_oublock;        /* Nombre d'écritures de blocs      */
  long  ru_msgsnd;         /* Nombre de messages émis          */
  long  ru_msgrcv;         /* Nombre de messages reçus         */
  long  ru_nsignals;       /* Nombre de signaux reçus          */
  long  ru_nvcsw;          /* Chgmnts de contexte volontaires  */
  long  ru_nivcsw;         /* Chgmnts de contexte involontaires*/
};
```

Signal funct

Signal	**sighandler_t signal(int** *signum*, **sighandler_t** *handler*)	Le comportement de **signal**() varie selon les versions d'Unix, et a également varié au cour du temps dans les différentes versions de Linux. **Évitez son utilisation** : utilisez plutôt sigaction.**signal**() installe le gestionnaire *handler* pour le signal *signum*. *handler* peut être **SIG_IGN**, **SIG_DFL** ou l'adresse d'une fonction définie par le programmeur http://manpagesfr.free.fr/man/man2/signal.2.html
Sigaction	**int sigaction(int** *signum*, **const struct sigaction** **act*, **struct sigaction** **oldact*)	L'appel système **sigaction**() sert à modifier l'action effectuée par un processus à la réception d'un signal spécifique. http://manpagesfr.free.fr/man/man2/sigaction.2.html
Kill	**int kill(pid_t** *pid*, **int** *sig*)	L'appel système **kill**() peut être utilisé pour envoyer n'importe quel signal à n'importe quel processus ou groupe de processus. http://manpagesfr.free.fr/man/man2/kill.2.html

- typedef void (*sighandler_t)(int) : pointeur sur fonction

```
- struct sigaction {
    void     (*sa_handler) (int);
    void     (*sa_sigaction) (int, siginfo_t *, void *);
    sigset_t   sa_mask;
    int        sa_flags;
    void     (*sa_restorer) (void);
}
- https://www-uxsup.csx.cam.ac.uk/courses/moved.Building/signals.pdf : signal list
```

Flow funct

Dup	**`int dup(int` *`oldfd`*`)`**	Créent une copie du descripteur de fichier *oldfd*. Utilise le plus petit numéro inutilisé pour le nouveau descripteur. http://manpagesfr.free.fr/man/man2/dup.2.html
Dup2	**`int dup2(int` *`oldfd`*`, int` *`newfd`*`)`**	Transforme *newfd* en une copie de *oldfd*, fermant auparavant *newfd* http://manpagesfr.free.fr/man/man2/dup.2.html
Pipe	**int pipe(int** *pipefd***[2])**	**pipe**() crée un tube, un canal unidirectionnel de données qui peut être utilisé pour la communication entre processus. http://manpagesfr.free.fr/man/man2/pipe.2.html

Directory funct

Chdir	**int chdir(const char **path*);**	remplace le répertoire de travail courant du processus appelant par celui indiqué dans le chemin *path*. http://manpagesfr.free.fr/man/man2/chdir.2.html
Opendir	**`DIR *opendir(const char *name);`**	La fonction **opendir**() ouvre un flux répertoire correspondant au répertoire *name*, et renvoie un pointeur sur ce flux. Le flux est positionné sur la première entrée du répertoire. http://manpagesfr.free.fr/man/man3/opendir.3.html
Closedir	**`int closedir(DIR *dir);`**	La fonction **closedir**() ferme le flux de répertoire associé à *dir*. Après cette invocation, le descripteur *dir* du flux de répertoire n'est plus disponible. http://manpagesfr.free.fr/man/man3/closedir.3.html
Readdir	**`struct dirent *readdir(DIR *dir);`**	La fonction **readdir**() renvoie un pointeur sur une structure *dirent* représentant l'entrée suivante du flux répertoire pointé par *dir*. Elle renvoie NULL à la fin du répertoire, ou en cas d'erreur. http://manpagesfr.free.fr/man/man3/readdir.3.html

Getcwd	**char *getcwd(char *buf, size_t size);**	La fonction **getcwd**() copie le chemin d'accès absolu du répertoire de travail courant dans la chaîne pointée par *buf*, qui est de longueur *size*. http://manpagesfr.free.fr/man/man3/getcwd.3.html

Sous Linux, la structure *dirent* est définie de la façon suivante :

```
struct dirent {
    ino_t          d_ino;       /* numéro d'inœud */
    off_t          d_off;       /* décalage jusqu'à la dirent suivante */
    unsigned short d_reclen;    /* longueur de cet enregistrement */
    unsigned char  d_type;      /* type du fichier */
    char           d_name[256]; /* nom du fichier */
};
```

https://pub.phyks.me/sdz/sdz/arcourir-les-dossiers-avec-dirent-h.html : lien utile

File funct

Stat	**int stat(const char** ***path*****, struct stat** ***buf*****);**	récupère l'état du fichier pointé par *path* et remplit le tampon *buf*. http://manpagesfr.free.fr/man/man2/stat.2.html
Lstat	**int fstat(int** *fd***, struct stat** ***buf*****);**	est identique à **stat**(), sauf que si *path* est un lien symbolique, il donne l'état du lien lui-même plutôt que celui du fichier visé. http://manpagesfr.free.fr/man/man2/stat.2.html
Fstat	**int lstat(const char** ***path*****, struct stat** ***buf*****);**	est identique à **stat**(), sauf que le fichier ouvert est pointé par le descripteur *fd*, obtenu avec open. http://manpagesfr.free.fr/man/man2/stat.2.html
Unlink	**int unlink(const char** ***pathname*****);**	**unlink**() détruit un nom dans le système de fichiers. Si ce nom était le dernier lien sur un fichier, et si aucun processus n'a ouvert ce fichier, ce dernier est effacé, et l'espace qu'il utilisait est rendu disponible. http://manpagesfr.free.fr/man/man2/unlink.2.html
Execve	**int execve(const char** ***fichier*****, char *const** *argv***[], char *const** *envp***[]);**	exécute le programme correspondant au *fichier*. Celui-ci doit peut être un exécutable binaire ou bien un script http://manpagesfr.free.fr/man/man2/execve.2.html
Access	**int access(const char** ***pathname*****, int** *mode***);**	**access**() vérifie si le processus appelant peut accéder au fichier *pathname*. http://manpagesfr.free.fr/man/man2/access.2.html

loctl	**int ioctl(int** *d*, **int** *requête*, **...);**	http://manpagesfr.free.fr/man/man2/ioctl.2.html

struct stat {

```
    dev_t     st_dev;      /* ID du périphérique contenant le fichier */
    ino_t     st_ino;      /* Numéro inœud */
    mode_t    st_mode;     /* Protection */
    nlink_t   st_nlink;    /* Nb liens matériels */
    uid_t     st_uid;      /* UID propriétaire */
    gid_t     st_gid;      /* GID propriétaire */
    dev_t     st_rdev;     /* ID périphérique (si fichier spécial) */
    off_t     st_size;     /* Taille totale en octets */
    blksize_t st_blksize;  /* Taille de bloc pour E/S */
    blkcnt_t  st_blocks;   /* Nombre de blocs alloués */
    time_t    st_atime;    /* Heure dernier accès */
    time_t    st_mtime;    /* Heure dernière modification */
    time_t    st_ctime;    /* Heure dernier changement état */
};
```

Terminal funct

Isatty	`int isatty(int fd);`	Renvoie 1 si *fd* est un descripteur de fichier ouvert connecté à un terminal, ou 0 autrement. https://man7.org/linux/man-pages/man3/isatty.3.html
Ttyname	`char *ttyname(int fd);`	La fonction **ttyname**() renvoie un pointeur sur le chemin d'accès terminé par un octet nul du périphérique terminal ouvert associé au descripteur de fichier *fd* http://manpagesfr.free.fr/man/man3/ttyname.3.html
Ttyslot	**int ttyslot(void)**	la fonction **ttyslot**() renvoie l'index du terminal de contrôle du processus appelant dans le fichier */etc/ttys* http://manpagesfr.free.fr/man/man3/ttyslot.3.html
Tcgetattr	`int tcgetattr(int fd, struct termios *termios_p);`	récupère les paramètres associés à l'objet référencé par *fd* et les stocke dans la structure *termios* pointée par *termios_p*. https://linux.die.net/man/3/tcgetattr

Tcsetattr	**int tcsetattr(int** *fd*, **int** *optional_actions*, **const struct termios** **termios_p*)	fixe les paramètres du terminal (à moins que le matériel sous-jacent ne le prenne pas en charge) en lisant la structure *termios* pointée par *termios_p*. *optional_actions* précise quand les changements auront lieu https://linux.die.net/man/3/tcsetattr

Plusieurs fonctions décrites ici utilisent un argument *termios_p* qui est un pointeur sur une structure *termios*. Cette structure contient au moins les membres suivants :

```
tcflag_t c_iflag;      /* modes d'entrée */
tcflag_t c_oflag;      /* modes de sortie */
tcflag_t c_cflag;      /* modes de contrôle */
tcflag_t c_lflag;      /* modes locaux */
cc_t     c_cc[NCCS];   /* caractères de contrôle */
```

Lien utile : http://manpagesfr.free.fr/man/man3/termios.3.html
https://manpages.debian.org/testing/manpages-fr-dev/tcsetattr.3.fr.html

Termcap library

Tgetent	**int tgetent(char *bp, const char *name);**	D'après la doc, cette fonction prend en premier paramètre l'adresse d'un buffer dans lequel stocker les informations utiles au fonctionnement interne de Termcap et en second paramètre le nom / le type de votre terminal.
Tgetflag	**int tgetflag(char *id);**	tgetnum permet de récupérer des informations numériques en rapport avec votre terminal. Comme par exemple le nombre de lignes et de colonnes.
Tgetnum	**int tgetnum(char *id);**	tgetflag fonctionne de la même manière que tgetnum à la différence prêt qu'il renvoie un booléen au lieu d'une valeur. Cette fonction est utilisée pour vérifier les capacités d'un terminal, savoir s'il est capable de faire tel ou tel action.
Tgetstr	**char *tgetstr(char *id, char **area);**	permet de récupérer les fameux termcaps sous la forme d'une séquence d'échappement ! On peut par exemple récupérer le termcap « cl » (pour **cl**ean) qui permet de nettoyer (vider) un terminal. tgetstr prend en deuxième paramètre l'adresse du buffer que l'on a utilisé pour tgetent
Tgoto	**char *tgoto(const char *cap, int col, int row);**	*Permet de déplacer le curseur, (à vérifier)*

Tputs	**int tputs(const char *str, int affcnt, int (*putc)(int));**	tputs est la fonction qui marche de pair avec tgetstr, c'est elle qui va se charger d'exécuter le termcap que l'on vient de récupérer.

Lien utile : https://zestedesavoir.com/tutoriels/1733/termcap-et-terminfo/

Liste termcap :

Termcap	Terminfo	Description
co	cols	nombre de colonnes affichées à l'écran
li	lines	nombre de lignes affichées à l'écran
AF	setaf	définit la couleur du texte
AB	setab	définit la couleur de fond
md	bold	affiche le texte en « gras »
us	smul	affiche le texte en « souligné »
mb	blink	affiche le texte en « clignotant »
cm	cup	déplace le curseur aux coordonnées souhaitées
cl	clear	efface le texte affiché à l'écran
me	sgr0	annule tous les changements opérés
os	os	*over strike*, précise si le terminal efface ou non le contenu lors d'une réécriture par dessus du texte (par exemple à la suite d'un *backspace*)

Utils

Printf	http://manpagesfr.free.fr/man/man3/printf.3.html
Malloc	http://manpagesfr.free.fr/man/man3/malloc.3.html
Free	http://manpagesfr.free.fr/man/man1/free.1.html
Open	http://manpagesfr.free.fr/man/man2/open.2.html
Read	http://manpagesfr.free.fr/man/man2/read.2.html
Write	http://manpagesfr.free.fr/man/man2/write.2.html
Close	http://manpagesfr.free.fr/man/man2/close.2.html
Exit	http://manpagesfr.free.fr/man/man3/exit.3.html
Strerror	http://manpagesfr.free.fr/man/man3/strerror.3.html
Perror	http://manpagesfr.free.fr/man/man3/perror.3.html