

Langage C

Wieslaw Zielonka
zielonka@irif.fr

convertir un chaîne de caractères en nombre

```
#include <stdlib.h>
```

```
double atof(const char *s)    convertit s en double  
int      atoi(const char *s)    en int  
long     atol(const char *s)    en long  
long long atoll(const char *s)  en long long
```

Ces fonctions ne prennent pas en compte des caractères d'espacement (dans le sens de `isspace()` (espace, '\n', '\t') au début de la chaîne et arrêtent la conversion quand elles rencontrent un caractère qui n'est pas un chiffre.

```
char *s = "    \n23abc  ";  
int i = atoi( s );           /* i == 23 */  
  
char t[] = "    \v  -54.89  mld";  
double d = atof( t );       /* d == -54.89 */  
  
int j = atoi("    \v\n  abc"); /* j == 0 */
```

les paramètres de main()

```
int main( int argc, char *argv[] ){ }
```

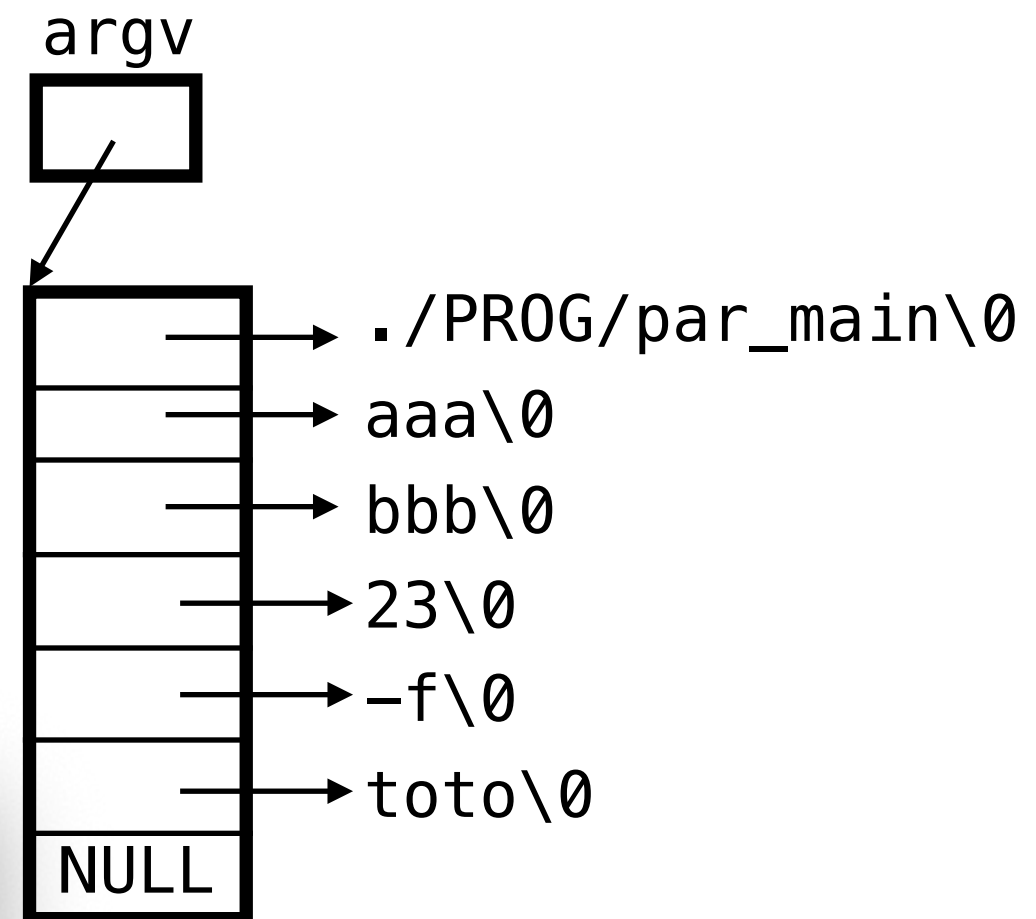
```
int main( int argc, char **argv ){ }
```

argc == 6

Supposons que le programme compilé `par_main` suivant se trouve dans le répertoire `PR0G`

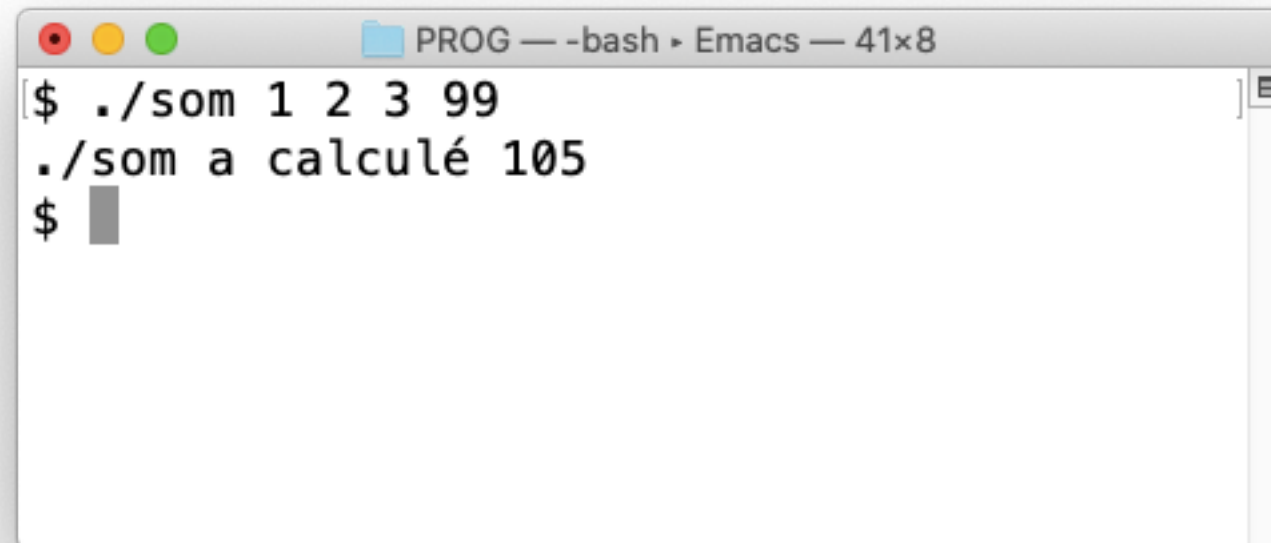
```
/* par_main.c */
#include <stdio.h>
int main(int argc, char *argv[]){
    for(int i = 0; i < argc; i++){
        printf("%s\n", argv[i]);
    }
}
```

```
COURS06 — -bash • Emacs — 49x9
$ ./PR0G/par_main aaa bbb 23 -f toto
./PR0G/par_main
aaa
bbb
23
-f
toto
$
```



paramètres de main()

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char **argv){
    int s = 0;
    for(int i = 1 ; i < argc; i++){
        s += atoi( argv[i] );
    }
    printf( "%s a calculé %d\n", argv[0] , s);
}
```

A terminal window titled "PROG — -bash > Emacs — 41x8" showing the execution of a program named 'som'. The user enters the command './som 1 2 3 99' and the program outputs './som a calculé 105'. The prompt '\$' is visible on the next line.

```
PROG — -bash > Emacs — 41x8
[$ ./som 1 2 3 99
./som a calculé 105
$
```

terminaison de programme

```
#include <stdlib.h>
```

```
void exit( int status )
```

La fonction `exit()` termine l'exécution de programme. Le paramètre : le code d'exit.

`exit(0)` – terminaison normale, `exit(i)` avec `i > 0` terminaison avec erreur ,

En pratique les valeurs `exit()` à utiliser entre 0 et 127.

```
int main(int arc , char **argv){  
    int i = ...  
    return i; /*    return i; dans main() le même effet que  
                *    exit(i)                                */  
}
```

(sauf si `main()` est appelé depuis une autre fonction ou récursivement -- oui c'est possible).

Mais on peut aussi quitter **`main()`** sans **`exit()`** ni **`return`**, juste parce qu'on a exécuté la dernière instruction de `main()`, dans ce cas `main()` termine avec la valeur de retour 0;
(Mais c'est valable uniquement pour `int main()` et pour aucune d'autre fonction qui retourne `int`.)

terminaison de programme

Comment voir le code de retour depuis le terminal? Après la terminaison de programme lancé depuis le terminal tapez sur le terminal

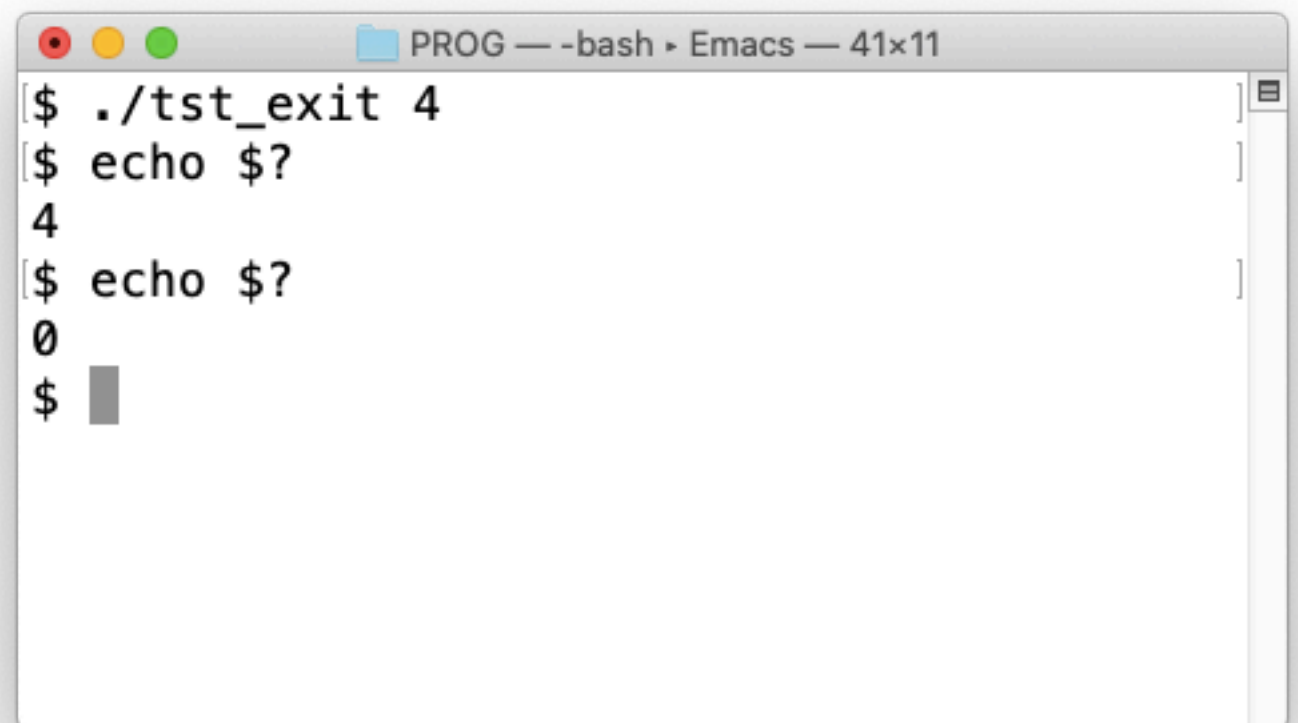
`echo $?`

Cela affiche la valeur de la variable `?` de `shell` : le code de retour de la dernière commande/programme C exécutée sur le terminal.

```
/* tst_exit.c */
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char **argv){
    if( argc != 2 ){
        printf("%s code\n", argv[0] );
        exit(0);
    }

    int a = atoi(argv[1]);
    exit(a);
}
```



```
PROG — -bash • Emacs — 41x11
$ ./tst_exit 4
$ echo $?
4
$ echo $?
0
$
```

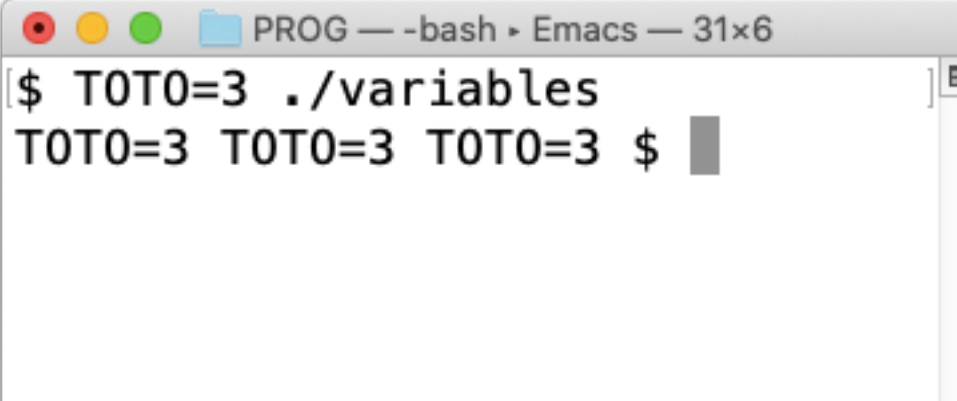
char *getenv(const char *name)

la fonction `getenv()` (`stdlib.h`) prend en paramètre le nom d'une variable d'environnement et retourne la valeur de cette variable.

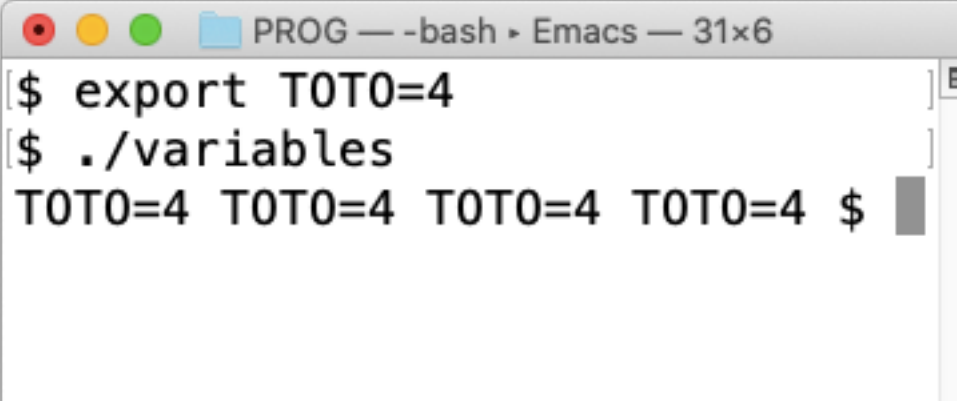
Exemple : `getenv("SHELL")`
retourne sur mon portable `"/bin/bash"`

```
/* variables.c */
#include <stdlib.h>
#include <std
int main(){
    char *var = getenv( "TOT0" );
    if( var == NULL )
        return 0;

    int d = atoi( var );
    for(int i = 0 ; i < d ; i++){
        printf("TOT0=%d ", d);
    }
}
```



```
PROG — -bash • Emacs — 31x6
$ TOT0=3 ./variables
TOT0=3 TOT0=3 TOT0=3 $
```



```
PROG — -bash • Emacs — 31x6
$ export TOT0=4
$ ./variables
TOT0=4 TOT0=4 TOT0=4 TOT0=4 $
```

chaînes de caractères littérales dans le texte du programme

```
int main(){

    char *t = "De deux choses l'une, ou le puits était vraiment"
        "bien profond, ou elle tombait bien doucement; car elle eut"
        "tout le loisir, dans sa chute, de regarder autour d'elle et"
        "de se demander avec étonnement ce qu'elle allait devenir."
        "D'abord elle regarda dans le fond du trou pour savoir où elle"
        "allait; mais il y faisait bien trop sombre pour y rien voir. "
        "Ensuite elle porta les yeux sur les parois du puits, et s'aperçut"
        "qu'elles étaient garnies d'armoires et d'étagères; ça et là, "
        "elle vit pendues à des clous des cartes géographiques et des images."
        "En passant elle prit sur un rayon un pot de confiture portant "
        "cette étiquette, \"MARMELADE D'ORANGES.\" Mais, à son grand regret,"
        "le pot était vide: elle n'osait le laisser tomber dans la "
        "crainte de tuer quelqu'un; aussi s'arrangea-t-elle de manière"
        "à le déposer en passant dans une des armoires.";

    printf("%s\n", t);

}

printf(" un très long format .... "

        " et la suite du même format ... ",

        i, j ,    ... );
```