< C >

Dorian.H Mekni

Les pointeurs

03 | 02 |2020

Question de mémoire

- Quand on rentre un int ou autre, on lui affecte une valeur.
- le système alloue de l'espace memoire à cette dernière à qui correspond une adresse.
- Cette allocation se passe dans la mémoire vive ou communément appelé RAM

Les pointeurs

- La premiere utilité d’un pointeur c’est de créer une fonction qui va modifier des données
- Le pointeur va permettre de pointer une autre variable
- Le pointeur est donc une variable qui contient l’adresse d’une autre variable
-

Pointons les points importants des pointeurs

```
/*

Pointeur : variable contenant l'addresse d'une autre variable

%d -> affiche une adresse de variable ou pointeur

    | VARIABLES |

    myVariable  : valeur de la variable
    &maVariable : adresse de la variable

    | POINTEURS |

    *myPointer   = NULL ou *myPointer = &maVariable

    monPointeur  : adresse de la variable pointée
    *monPointeur : valeur de la variable pointrée
    &monPointeur : adresse du pointeur

 */
```

Localisation d’adresse

Pour découvrir l’adresse que le système d’exploitation a alloué à une variable on opérera manipulation la suivante:

```
int adresse = 18
printf( “il s’agit de  %p\n”, &adresse)
```

Créer un pointeur

- Lui ajouter une étoile comme ceci:
  *mon_premier_pointeur
- Un pointeur permet de faire une référence sur une valeur
- Il permet de modifier les données d’une fonction puisqu’il localise directement la variable et s’approprie de sa valeur à l’adresse de celle-ci.
-

reverse_numbers

```c
void reverse_numbers(int *numberA, int *numberB)

{
    int temporary = 0;

    temporary = *numberB;
    *numberB = *numberA;
    *numberA = temporary;
}
int main(void)

{
    int numberA = 10;
    int numberB = 20;

    printf("Number A = %d and Number B = %d\n", numberA, numberB);
    reverse_numbers(&numberA, &numberB);
    printf("Number A = %d and Number B = %d\n", numberA, numberB);


    return 0;
}
```

```
Number A = 10 and Number B = 20
Number A = 20 and Number B = 10
Program ended with exit code: 0
```

Initialisation de pointeur

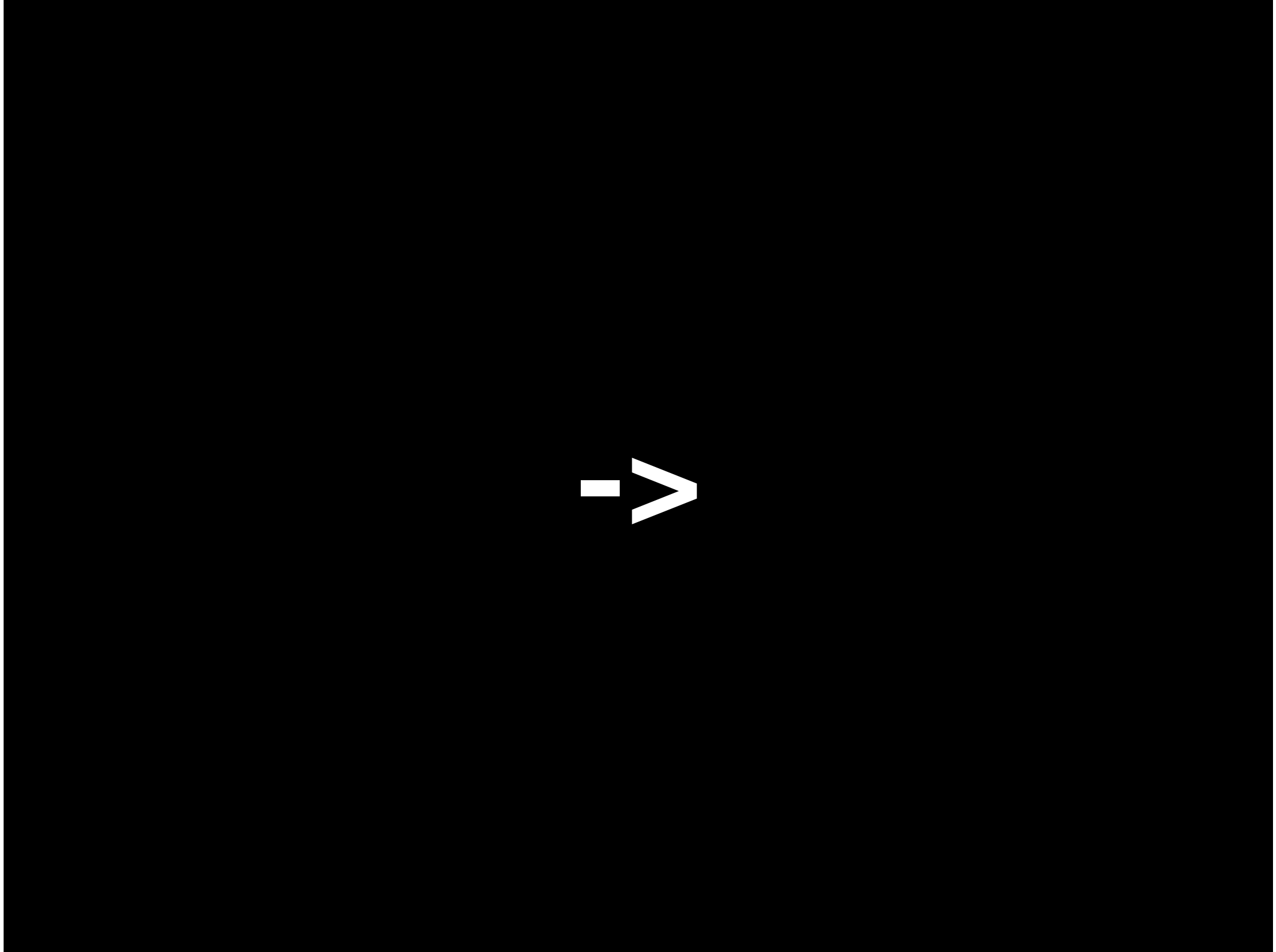
```c
void reverse_numbers(int *numberA, int *numberB)

{
    int temporary = 0;

    temporary = *numberB;
    *numberB = *numberA;
    *numberA = temporary;
}
int main(void)

{
    int numberA = 10;
    int numberB = 20;

    // Maintenant pour initialiser des pointeurs:
    int *ptrsOnNumberA = &numberA;
    int *ptrsOnNumberB = &numberB;

    printf("Number A = %d and Number B = %d\n", numberA, numberB);
    reverse_numbers(ptrsOnNumberA, ptrsOnNumberB);
    printf("Number A = %d and Number B = %d\n", numberA, numberB);


    return 0;
}
```

```
Number A = 10 and Number B = 20
Number A = 20 and Number B = 10
Program ended with exit code: 0
```

Un dernier pour la route

```c
void change_number(int *number)

{

    *number = 11;
}
int main(void)

{
    int random = 5;



    printf("Number = %d\n", random);
    change_number(&random);
    printf("Number = %d\n", random);


    return 0;
}
```

```
Number = 5
Number = 11
Program ended with exit code: 0
```


->
->
->