Langage C

Wieslaw Zielonka
zielonka@irif.fr

variables static

Rappel :

une variable globale ou un tableau global avec l'attribut static sont visible uniquement à l'intérieur de fichier qui les déclare.

static = local (à un fichier) et permanent

variables static dans une fonction

Une variable `static` est permanente et réside dans une zone de mémoire différente de la pile et du tas.

```
int compteur(void){

  static int c = 9;

  c++;

  return c;

}

int main(void){

  for(int i = 0; i < 8; i++){

    printf("compteur = %d\n",
            compteur() );

  }

  return 0;

}
```

sur le terminal :

```
compteur = 10
compteur = 11
compteur = 12
compteur = 13
compteur = 14
compteur = 15
```

Une variable static déclarée dans une fonction est une variable permanente mais visible uniquement à l'intérieur de la fonction.

Elle est initialisée une seule fois : avant le début de l'exécution de programme.

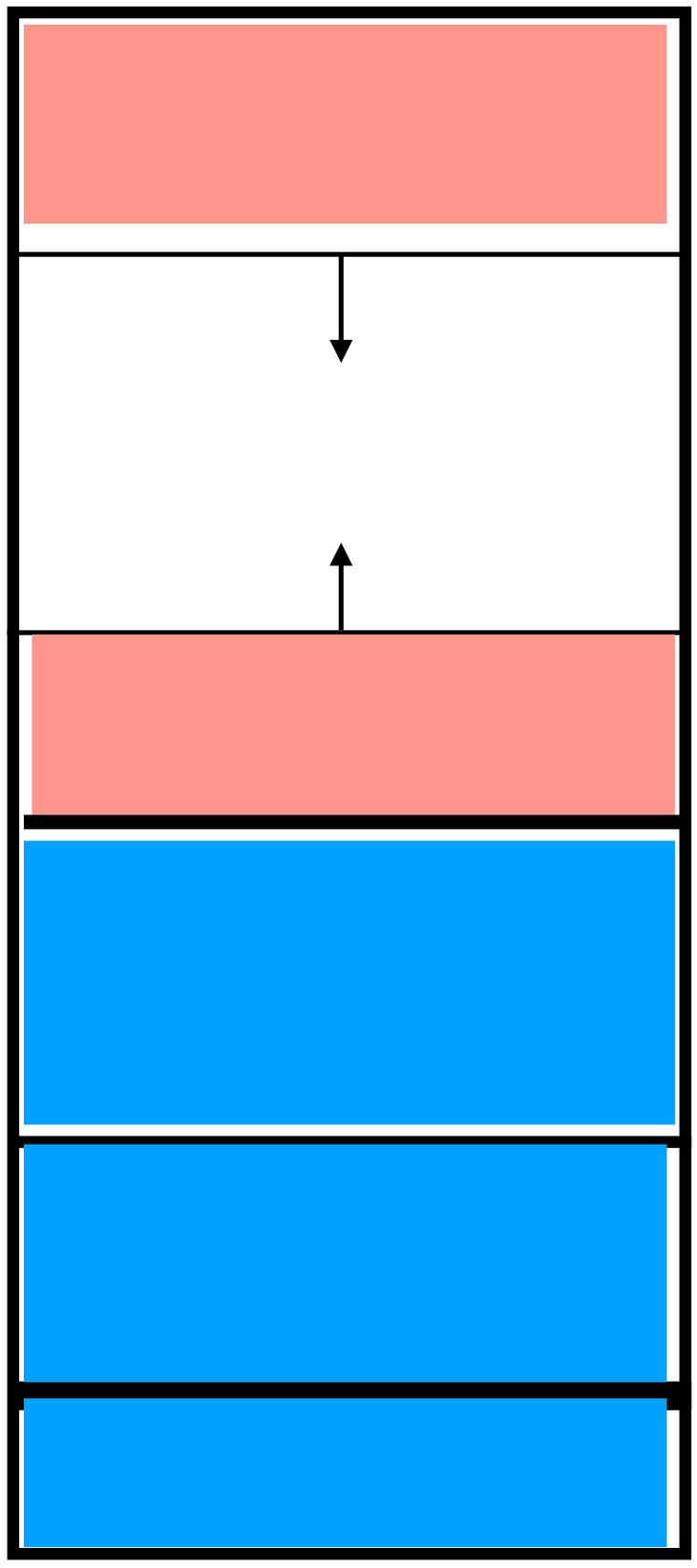
variables static dans une fonction

```
int compteur(void){
  static int c = 9;
  c++;
  return c;
}
int main(void){
  for(int i = 0; i < 8; i++){
    printf("compteur = %d\n",
            compteur() );
  }
  return 0;
}
```

```
static int c = 9;
int compteur(void){
   c++;
  return c;
}
int main(void){
  for(int i = 0; i < 8; i++){
    printf("compteur = %d\n",
            compteur() );
  }
  return 0;
}
```

La seule différence entre les deux programmes est que dans le programme à gauche c est visible uniquement à l'intérieur de la fonction compteur() tandis que à droit c est visible dans tous le fichier.

mémoire d'un programme



pile : variables et tableaux non-static déclarés dans une fonction.

mémoire permanente

mémoire non-permanente

tas : mémoire de malloc

variables et tableaux de niveau 0 noninitialisés
et variables et tableaux static noninitialisées (de tous niveau). Initialisation par défaut à 0.

variables et tableaux de niveau 0 initialisés
et variables et tableaux static initialisées (de tous niveau)

text : le code du programme

static - exemple

```
char *supprimer_espaces(const char *s){

#define WORD_SIZE 100

  static char mot[WORD_SIZE];

  int i = 0;

  while( i < WORD_SIZE-1 & *s != '\0' ){

    if( !isspace( *s ) )
       mot[i++] = *s;

    s++;
  }

  mot[i] = '\0';

  return mot;

}

int main(void){

  char *a = supprimer_espaces("ABRA KADA B R A");

  printf("%s\n",a);          ->  ABRAKADABRA

  char *b= supprimer_espaces("ALA MA KOTA");

  printf("%s\n",b);          ->  ALAMAKOTA

}
```

`supprimer_espaces()`
retourne un pointeur vers une chaîne de caractères
qui contient tous les caractères de la chaîne s sauf les espaces

static - exemple

```
char *supprimer_espaces(const char *s){

#define WORD_SIZE 100

  char mot[WORD_SIZE];

  int i = 0;

  while( i < WORD_SIZE-1 & *s != '\0' ){

    if( !isspace( *s ) )
       mot[i++] = *s;

    s++;

  }

  mot[i] = '\0';

  return mot;

}

int main(void){

  char *a = supprimer_espaces("ABRA KADA B R A");

  printf("%s\n",a);          ->  ABRAKADABRA

  char *b= supprimer_espaces("ALA MA KOTA");

  printf("%s\n",b);          ->  ALAMAKOTA

}
```

même chose sans static : incorrect.

La fonction retourne un pointeur vers un tableau sur la pile qui n'est plus valable après return.