Bienvenue au cours de C - SUPINFO 2022

Présentation et déroulement du cours

Anthony Rabine

Freelance C/C++ embarqué et Qt/QML

Table of contents

Je me présente

Je me présente

Je m'appelle **Anthony Rabine** et je suis développeur Freelance en C/C++ depuis 2018, plutôt à orientation industrielle :

- Programmation embarqué (microcontrôleurs, Arduino, Raspberry PI)
- Programmation Qt/QML (kiosques interactifs, bancs de production, outillage, transports...)

Mon parcours

J'ai suivi un parcours universitaire et professionnel cohérent : de l'électronique et de la programmation embarquée, tout le temps.

- **1998 - 2003** → DUT, Licence, Maîtrise, DESS (équivalent Master 2) en GEII (Génie Électrique et Informatique Industrielle)
- **2004 - 2009** → Ingénieur d'études (Chauvin-Arnoux, Paris), développeur embarqué C, C++ sur DSP et Atmel 8 bits (domaine : comptage électrique)
- **2010 - 2018** → Ingénieur senior / architecte embarqué (Itron), développeur C++ sur compteurs électriques dont le Linky, suivi de projet, management d'équipe de développeurs
- Depuis 2018 → Freelance

Un petit mot sur le C

Usage du C dans la vraie vie

→ Le C est classiquement le langage de base des applications systèmes (OS) et est l'inspiration de nombreux autres langages (surtout au niveau de sa syntaxe)

→ Tous les logiciels autour de vous sont développés au moins en partie en C

→ Autrefois, beaucoup de jeux vidéo aussi, maintenant beaucoup mois mais certaines API restent en C (OpenGL par exemple)

→ Le développement embarqué bascule petit à petit vers du C++ mais le C reste le langage de référence. Beaucoup de librairies ne sont fournies qu'en C.

Le C évolue lentement

Le C est un standard ISO (son comportement est décrit) relativement conservateur et aux évolutions lentes. Sa force réside par sa simplicité, sa portabilité les librairies tierces nombreuses.

- C99 : la référence actuellement (ce que l'on trouve le plus)
- C11 : multithreading standardisé, macros, génériques, fonctions obsolètes supprimées ...
- C17 : corrections, aucune nouveauté dans le langage

De plus, les compilateurs sont maintenant performants, facile d'usage et affichent des messages d'erreur plus descriptifs (GCC / Clang).

Conseils

- Il est important de connaître le C, rien que pour sa culture générale
- L'année prochaine, vous aurez du C++ au programme, il faut garder en tête le C, la base est la même
- Utilisez l'IDE de votre choix (CLion, Code::Lite, Visual Studio Code, VIM...)

Conclusion

Contact

Mon CV : `rabine.fr`

Mon GitHub : `rabine.fr`

Mon site corpo : `d8s.eu`

Mail : `anthony@d8s.eu`

Bugs C dans des logiciels Open Source célèbres

Quake-III-Arena

```
static ID_INLINE int BigLong(int l)
{ LongSwap(l); }
```

Quake-III-Arena

Ici il manque un return à la fin de la fonction, probablement return LongSwap(l);. Le résultat est inconnu, cela dépend de la valeur d'un registre du CPU où est normalement stocké la valeur de retour.

Cette erreur a été détectée par un logiciel d'analyse de code statique.

Quake-III-Arena

```
void Item_Paint(itemDef_t *item) {
  vec4_t red;
  menuDef_t *parent = (menuDef_t*)item->parent;
  red[0] = red[3] = 1;
  red[1] = red[2] = 0;
  if (item == NULL) {
    return;
  }
  ...
}
```

Quake-III-Arena

Le pointeur item est testé après son utilisation ; il faut bien évidemment tester son éventuelle nullité avant de l'utiliser. Typiquement une modification de fonction réalisée après sont implémentation d'origine.

Cette erreur a été détectée par un logiciel d'analyse de code statique.

Apache HTTP server

```
typedef  size_t       apr_size_t;
APU_DECLARE(apr_status_t) apr_memcache_getp(...)
{
  ...
  apr_size_t len = 0;
  ...
  len = atoi(length);
  ...
  if (len < 0) {
    *new_length = 0;
    *baton = NULL;
  }
  else {
    ...
  }
}
```

Apache HTTP server

La variable len est de type non signée ; dès lors, elle ne peut jamais être négative et donc le test if (len < 0) est toujours faux.

Cette erreur a été détectée par un logiciel d'analyse de code statique.

Miranda IM

```
static char *_skipblank(char * str)
{
  char * endstr=str+strlen(str);
  while ((*str==' ' || *str=='\t') && str!='\0') str++;
  while ((*endstr==' ' || *endstr=='\t') &&
         endstr!='\0' && endstr<str)
    endstr--;
  ...
}
```

Miranda IM

La condition de sortie de la boucle risque de poser problème !
L'auteur du code a probablement voulu écrire `*str != '0'`. Ici le pointeur est important car on veut tester la valeur d'un caractère.

Cette erreur a été détectée par un logiciel d'analyse de code statique.

Chromium

```
void Time::Explode(..., Exploded* exploded) const {
  ...
  ZeroMemory(exploded, sizeof(exploded));
  ...
}
```

Chromium

Attention à ce que vous passez à l'opérateur sizeof() ! Ici, on passe un pointeur, donc on obtiendra la taille d'un pointeur qui est de 4 ou 8 octets selon la machine. L'auteur voulait effacer tout le buffer, donc il faut préciser l'objet à sizeof() pour qu'il calcule la bonne taille : ZeroMemory(exploded, sizeof(*exploded));.

Cette erreur a été détectée par un logiciel d'analyse de code statique.

Chromium

```
void MD5::finalize () {
  ...
  uint1 buffer[64];
  ...
  // Zeroize sensitive information
  memset (buffer, 0, sizeof(*buffer));
  ...
}
```

Chromium

Ici, l'opérateur sizeof() va calculer la taille du premier élément du tableau. Pour obtenir la taille du tableau, la bonne écriture est :

`memset(buffer, 0, sizeof(buffer));`.

Cette erreur a été détectée par un logiciel d'analyse de code statique.

BCmenu

```
void BCMenu::InsertSpaces(void)
{
  if(IsLunaMenuStyle())
    if(!xp_space_accelerators) return;
  else
    if(!original_space_accelerators) return;
  ...
}
```

BCmenu

Il manque des accolades ! Il n'est pas évident que le else appartienne au premier if … ce n'est d'ailleurs pas le cas, le compilateur va associer le else au plus proche if, c'est-à-dire le deuxième.

Apple SSL library

```
. . .
hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
hashOut.length = SSL_SHA1_DIGEST_LEN;
if ((err = SSLFreeBuffer(&hashCtx)) != 0)
    goto fail;
if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
    goto fail;
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;

err = sslRawVerify(...);
. . .
```

Apple SSL library

Le fameux bug d'Apple dans sa librairie SSL... le deuxième goto fail saute la vérification du certificat SSL qui pourra être, dans certains cas, accepté.

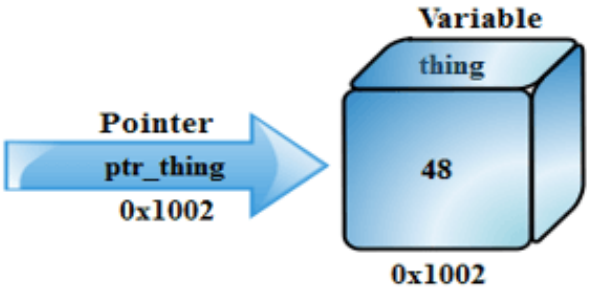
Apple SSL library

Le fameux bug d'Apple dans sa librairie SSL... le deuxième goto fail saute la vérification du certificat SSL qui pourra être, dans certains cas, accepté.

Illustrations sur le fonctionnement des pointeurs

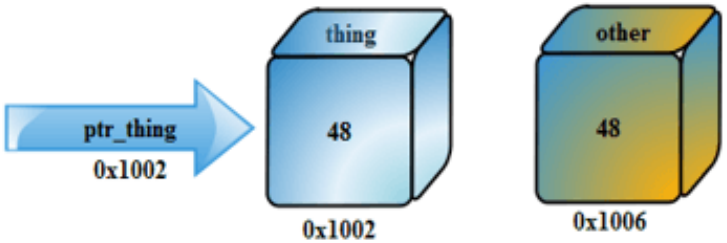
Représentation des pointeurs 1

A) ptr_thing = &thing;



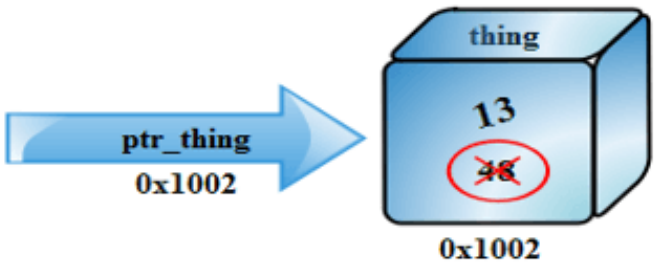
Assigns thing's address to ptr_thing

B) other = *ptr_thing;



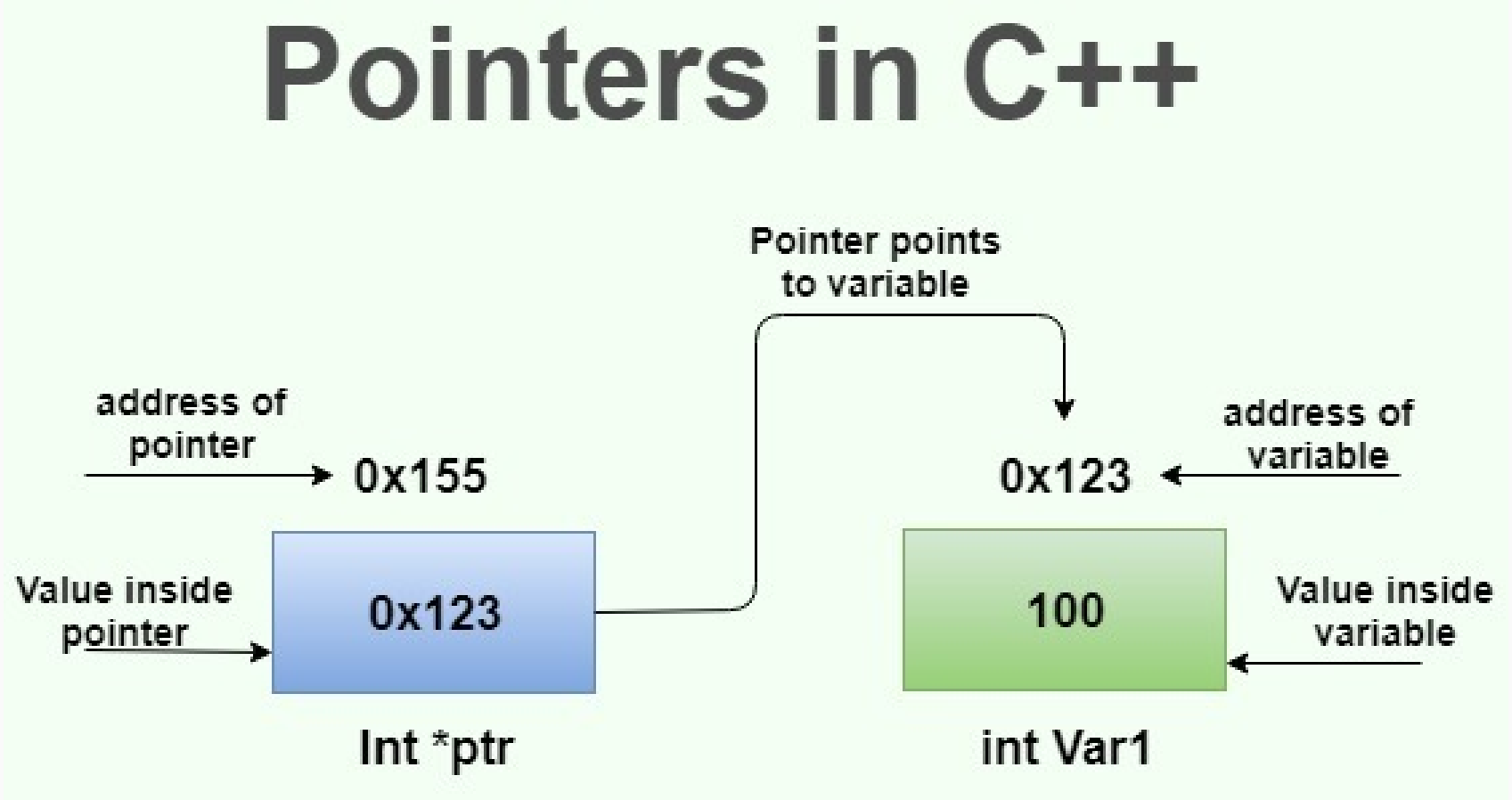
Assigns to other the value pointed by ptr_thing

C) *ptr_thing=13;

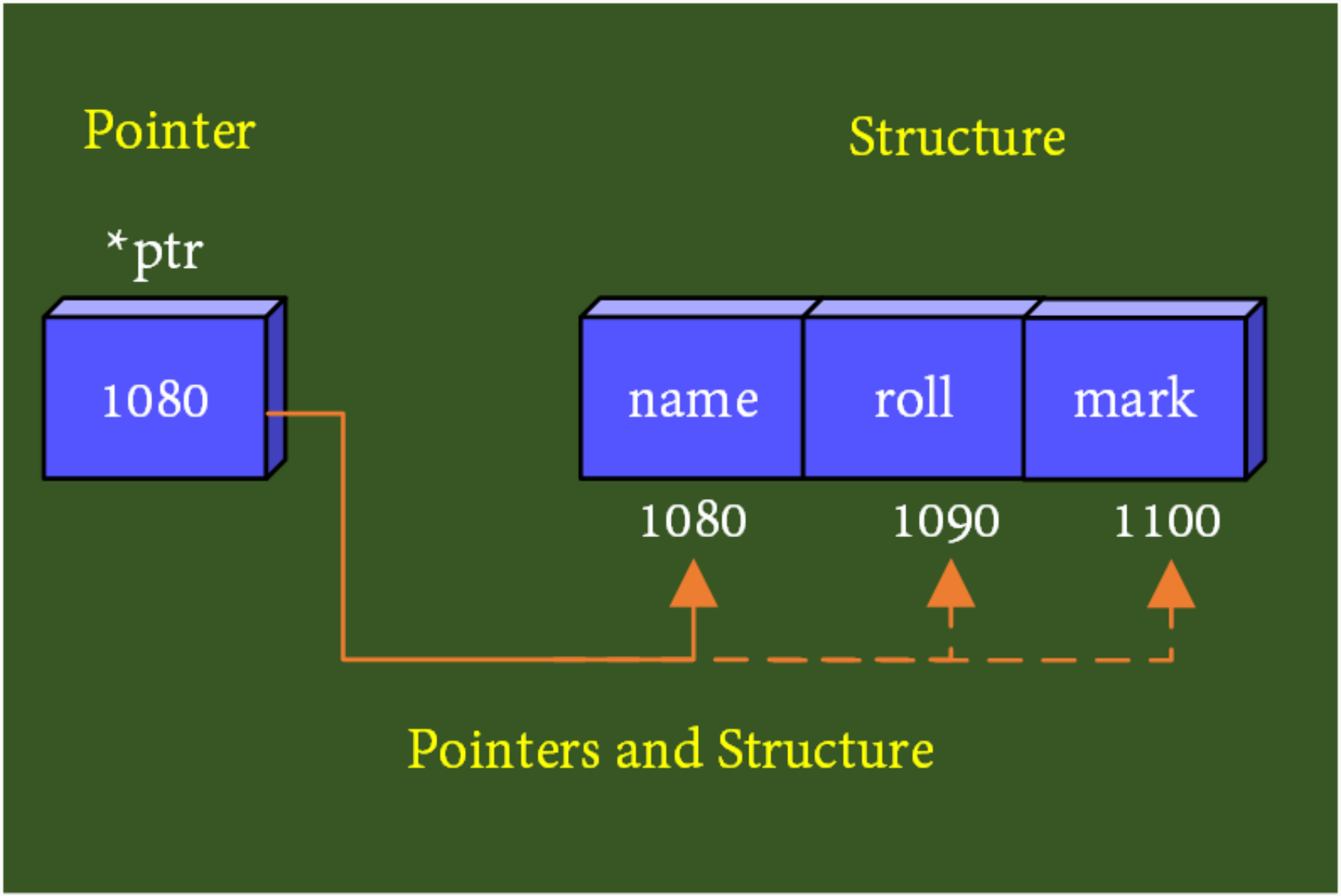


Assigns new value to the pointed variable

Représentation des pointeurs 2



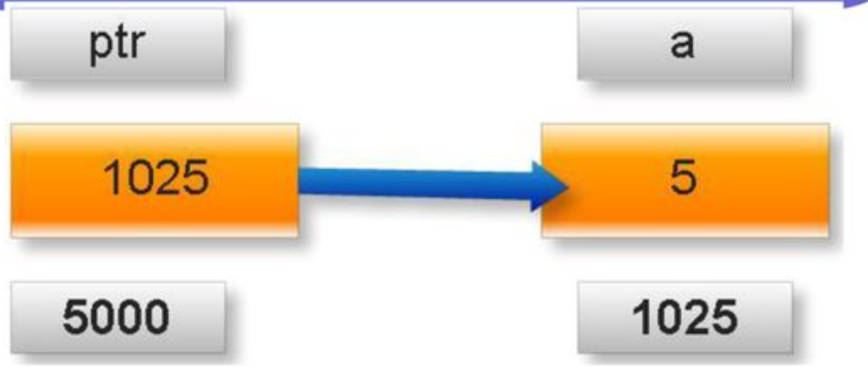
Représentation des pointeurs 3



Représentation des pointeurs 4

Pointers

```
int a=5;
int * ptr;
ptr=&a;
```



About variable a:

- 1. Name of variable : a
- 2. Value of variable which it keeps: 5
- 3. Address where it has stored in memory : 1025 (assume)

About variable ptr:

- 4. Name of variable : ptr
- 5. Value of variable which it keeps: 1025
- 6. Address where it has stored in memory : 5000 (assume)

Quelques règles de style et codage

Général

En règle générale évitez les majuscules dans le nom des fichiers pour éviter des problèmes entre les OS Windows et Unix.

Fichiers de module (.h et .c) : tout en minuscules, mots séparés par le caractère underscore. Variables : minuscules, mots séparés par le caractère underscore.

Accolades et constatntes

```
if (depth_in_ft > 10) dive_stage = DIVE_DEEP; // This is legal...
else if (depth_in_ft > 0)
  dive_stage = DIVE_SHALLOW; // ... as is this.
else
{ // But using braces is always safer.
  dive_stage = DIVE_SURFACE;
}
```

Toujours utiliser des des accolades pour les if, else, switch, while, do, for. De préférence en colonne, de façon à bien voir l'ouverture et la fermeture.

Parenthèse autour des forumules

```
if ((depth_in_cm > 0) && (depth_in_cm < MAX_DEPTH))
{
  depth_in_ft = convert_depth_to_ft(depth_in_cm);
}
```

Ne pas se reposer sur l'ordre des opérateurs du langage C; rendez votre code explicite, et on implicite. Ceci est vrai pour toutes les formules mathématiques, utilisez des parenthèses.

Du bon usage de static et const

```c
static const ma_struct private_variable = {...}; //  dans le .c
```

Réfléchissez sur la portée de chaque variable : est-elle publique ? privée ? globale au module ou à tout le logiciel . Renforcez les variables globales à un module par le mot-clé static. Si c'est une constante, ajoutez const.

Structures

```
typedef struct
{
uint16_t count;
uint16_t max_count;
uint16_t control;
} timer_reg_t;
```

Voici comment nommer vos structures. Le suffixe _t signifie type. C'est votre nouveau type à vous. De préférence utilisez les types des entiers fournis dans stdint.h pour rendre votre code le plus portable possible sur différents OS et CPU.

Espaces et commentaires

```
// Step 1: Batten down the hatches.
for (int hatch = 0; hatch < NUM_HATCHES; hatch++)
{
  if (hatch_is_open(hatches[hatch]))
  {
    hatch_close(hatches[hatch]);
  }
}
// Step 2: Raise the mizzenmas
// ...
```

Commentez de manière intelligente : décrivez les étapes, le code difficile. Le reste est trivial à la lecture du code. Éclaircissez vos mots clés et l'intérieur des parenthèses : espaces avant et après les opérateurs, espace après le mot clé…

Defines

```
#ifdef USE_UNICODE_STRINGS
# define BUFFER_BYTES 128
#else
# define BUFFER_BYTES 64
#endif
```

Vos macros doivent être en majuscule. Il est possible d'indenter les directives du pré-processeur en ajoutant des espace après le caractère dièze.