



5.0

Nom : Carnevale

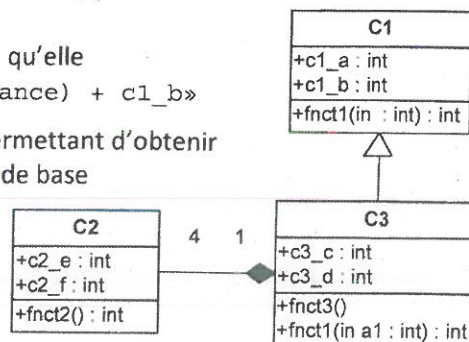
Prénom : Antonio

Classe : T/2

Date : 17.06.2011

Problème n° 1 (programmation orienté-objet)

1. Déclarez complètement les structures C pour le diagramme de classes ci-dessous. La classe C3 surcharge la fonction «fnc1» de la classe C1.
2. Implémentez la fonction «fnc1» de la classe C3 de manière à ce qu'elle retourne la somme de «a1 + c3_d + c2_e (de la 2^{ème} instance) + c1_b»
3. Implémentez les macros «offset_of» et «container_of» permettant d'obtenir la référence sur l'objet dérivé à partir de la référence sur la classe de base



c1.h

```
struct class_c1_t {
    int c1_a;
    int c1_b;
    int (*fnc1)(struct class_c1_t* c1_t, int in);
};
```

```
extern struct class_c1_t* c1_init(struct class_c1_t* c1_t);
```

c1.c

#include c1.h

```
struct class_c1_t* c1_init(struct class_c1_t* c1_t) {
```

```
    c1_t->c1_a = c1_a;
```

```
    c1_t->c1_b = c1_b;
```

```
    c1_t->fnc1 = c1_fnc1;
```

```
    return c1_t;
}
```

```
static int c1_fnc1(struct class_c1_t* c1_t, int in) {
```

```
    ;
}
```

c2.h

```
#include c1.h
#include c2.h
```

```
struct class_c2_t {
```

```
    struct class_c1_t m_base;
```

```
    int c2_c;    int c2_d;
```

```
    int (*fnc3)(struct class_c2_t* c2_t);
```

```
extern struct class_c1_t c2_init(struct class_c1_t* c1_t,
```

C3.c

#include C1.h
#include C2.h
#include C3.h

struct class.c3.t* c3_init(struct class.c3.t* c3.t) {

c3.t->m_base = c3.t;

c3.t->m_base.funct1 = c3.funct1; // overloading

c3.t->c3.e = c3.e;

c3.t->c3.d = c3.d;

c3.t->c3.funct3 = c3.funct3;

return &c3.t->m_base;

c3.t->m_base.c1.e = c3.c1.e;

c3.t->c2.t.c2.e = c3.c2.e;

c3.t->c2.t.c2.f = c3.c2.f;

c3.t->c2.t.funct2 = c3.funct2;

pos
base
offset
combined

static int c3.funct1(struct class.c3.t* c3.t, int a1) {

struct class.c3.t* c3.t = container_of(c3.t, struct class.c3.t, m_base);

return a1 + c3.d + c3.c2.e + c3.c1.e;

↑
equivalent to c3.e

↑
equivalent to c3.b

#define offset_of
011 (type, member)
...
not a macro?

? offset_of(class.c3.t, m_base);

C2.h

struct class.c2.t {

int c2.e; /* c2.t */

int (*funct2)(struct class.c2.t* c2.t);

return void c2_init(struct class.c2.t* c2.t);

C2.c

#include C2.h

void c2_init(struct class.c2.t* c2.t) {

c2.t->c2.e = c2.e;

c2.t->c2.f = c2.f;

c2.t->c2.funct2 = c2.funct2;

int c2_funct2(struct class.c2.t* c2.t) {

}

Microprocesseurs 1 & 2: Travail écrit no 4.

Problème n° 2 (Toolchain)

1. Concevez un Makefile pour la génération d'une application composée de 3 fichiers (main.c, file_a.c et file_b.c). Le programme exécutable sera appelé «appl». Le compilateur «gcc» sera utilisé pour la génération de l'application avec les flags de compilation «-W -Wall -Wextra -O2 -std=c99». Le Makefile devra également permettre d'effacer tous les fichiers générés. Il est impératif d'utiliser des variables pour spécifier les flags de compilation et les fichiers sources.

Makefile :

EXEC = appl
SRCS += main.c file_a.c file_b.c
CC = gcc

CFLAGS = -W -Wall -Wextra -O2 -std=c99
OBJS = \$(SRCS:.c=.o)

%.o :
gcc -c \$(CFLAGS) \$(OBJS) -o \$@

all : \$(EXEC)
\$(EXEC) : \$(OBJS)
ld \$(OBJS) \$(LDFLAGS) -o \$@

clean :
rm -rf \$(EXEC) \$(OBJS)
PHONY : all clean

2. Quel est la fonction des utilitaires suivants :

a. gdb

L'utilitaire "gdb" va nous permettre de lancer le débogage. (comme gcc qui indique le compilateur à utiliser)

b. gprof

L'utilitaire "gprof" va nous permettre d'analyser le profil de notre application. (profilage des données).

c. as

L'utilitaire "as" va nous permettre d'assembler le code en machine.

d. ld

L'utilitaire "ld" va nous permettre de lier les fichiers .o.

Microprocesseurs 1 & 2: Travail écrit no 4.

Problème n° 3 (Vérification)

1. Citez 3 techniques/méthodes permettant de valider des applications logicielles dans les différentes phases de leur développement

reviews / test unitaires & test système / test autonome test répétitifs
early review / CU-ASSERT, gcc... / la débogue un programme
constructive review / test à la fin d'un projet
+ Trig-label

2. Décrivez une technique/méthode permettant de garantir qu'un composant logiciel a été correctement et si possible complètement vérifié. Citez un utilitaire de la chaîne d'outils GNU permettant de mettre d'utiliser cette méthode/technique ainsi que la façon de le mettre en œuvre.

gcc ou bien faire pointer CU-ASSERT

pour activer les CU-ASSERT à l'aide de la méthode de compilation

3. Implémentez, pour la fonction « memchr » de la librairie standard C (selon description ci-dessous), un test unitaire permettant de valider/vérifier deux résultats positifs et deux résultats négatifs.

```
/**
 * Locate character in block of memory
 * Searches within the first num bytes of the block of memory pointed by ptr for the
 * first occurrence of value (interpreted as an unsigned char), and returns a pointer
 * to it.
 * @param ptr pointer to the block of memory where the search is performed.
 * @param value value to be located. The value is passed as an int, but the function
 * performs a byte per byte search using the unsigned char conversion of
 * this value.
 * @param num number of bytes to be analyzed.
 *
 * @return A pointer to the first occurrence of value in the block of memory pointed
 * by ptr. If the value is not found, the function returns NULL.
 */
void * memchr (const void * ptr, int value, size_t num );
```

CU-ASSERT (memchr (0x00001000, 1, 10) == 0x00001000)
CU-ASSERT (memchr (0x00001000, 2, 10) == 0x00001001)
CU-ASSERT (memchr (0x00001000, -1, 10) == 0x00001000)
CU-ASSERT (memchr (0x00001000, 7, 10) == 0x00001000)

for test2 - very

Microprocesseurs 1 & 2: Travail écrit no 4.

Problème n° 4 (Documentation)

1. Indiquez 4 outils permettant de documenter du logiciel.

- Conception (analyse, conception, design, ...)
- Spécifications (contraintes, bugs, ...)
- Code (commentaires, source code management, ...)
- Aides (Wiki, manuel de l'utilisateur, ...)

2. Citez 2 outils permettant de simplifier le développement de logiciels et d'améliorer sa qualité

- Automate Build Tools (Makefile, ...)
- Source Code Management (SVN, ...)

3. Indiquez une manière de structurer le logiciel et sa documentation afin de simplifier son développement et sa maintenance



4. Décrivez succinctement comment on peut procéder pour réviser du logiciel

On peut utiliser les Tag/Label : **+SEM**

→ Label de module : + grande granularité

→ Label de composants : + simple à implémenter

5. Citez les 3 niveaux principaux de la documentation du logiciel (public cible)

- Documentation de l'utilisateur
- Documentation du développeur
- Documentation du système

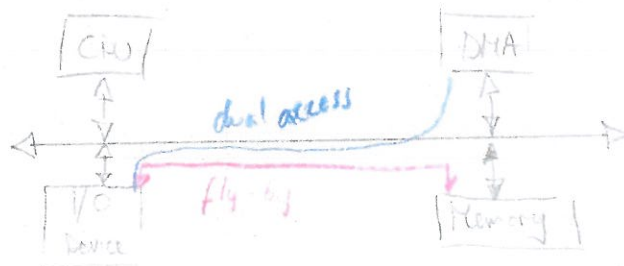
Microprocesseurs 1 & 2: Travail écrit no 4.

Problème n° 5 (DMA)

1. Décrivez succinctement le principe d'un DMA.

Le contrôleur DMA devra avoir accès aux signaux des bus
contrôle, adresse et données, ceci afin de décharger le CPU
de ses tâches (c'est à dire qu'il va "prendre les transactions" sur ces bus).

2. Décrivez à l'aide d'une figure l'architecture DMA



Le rôle du contrôleur DMA est de contrôler le transfert de données entre
l'I/O et la Memory.

3. Citez les 4 phases principales d'un transfert DMA entre une mémoire et un périphérique



4. Citez 2 modes d'opérations des DMA du processeur i.MX27

Il y a 3 modes d'opérations, voici 2 de ceux-ci:

Linear memory: transfert de données entre la mémoire et un périphérique (autre)
FIFO: transfert de données entre la mémoire FIFO et un périphérique (autre)

5. Citez 2 exemples d'application avec DMA

Echantillonnage de données à intervalles réguliers

- Contrôleurs ethernet
- Contrôleurs bus PCI
- Contrôleurs disques durs (IDE, SATA, ...)