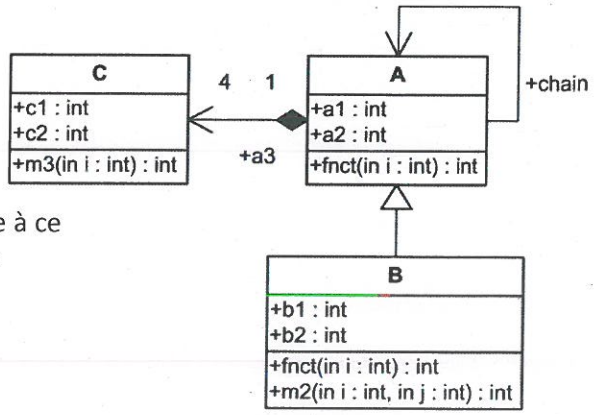
Microprocesseurs 1 & 2: Travail écrit no 4. 4.8

Nom : ZOPPI **Prénom :** DAMIANO **Classe : T/2** **Date : 11.06.2012**

5

Problème n° 1 (programmation orienté-objet)

1. Pour le diagramme de classes ci-contre :
   a. Déclarez les classes A, B et C en langage C orienté-objet.
   Remarque : la classe B surcharge la fonction «`fnct`» de la classe A.
   b. Implémentez la fonction «`fnct`» de la classe B de manière à ce qu'elle retourne la somme de « `i + B::b1 + A::a2` »




```
struct A {
    int a1;
    int a2;
    int (*fnct)(struct A*, int i);
} struct C a3[4];

struct B {
    int b1;
    int b2;
    int (*m2)(struct B*, int i, int j)
```

```
struct C {
    int c1;
    int c2;
    int (*m3)(struct C*, int);
};
struct A {
    int a1;
    int a2;
    struct C a3[4];
    struct A* chain;
    int (*fnct)(struct A*, int);
```

```
int B fnct(struct A* ouf a, int i) {
    struct B* ouf = container_of(ouf a, struct B, B2sc_2)
    ...
}
```

2. 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.

```
# define offset_of(type, member)
((char*) (&(type*)0)-> member)
```

```
# define container_of (ptr, type, member)
(char*)((ptr*)-offset_of (type, member))
```

3. Décrivez succinctement le principe d'orienté-objet en langage C.

En C les objets ne sont pas supportés. Pour créer des objets il faut utiliser une structure contenant les attributs qui définissent l'état de l'objet. Il faut également y ajouter les méthodes pour opérer sur l'objet (pour modifier les attributs) qui se trouvent des pointeurs de fonctions. Chaque méthode doit avoir en paramètre la référence à l'objet (structure)

Microprocesseurs 1 & 2: Travail écrit no 4.

8

Problème n° 2 (Toolchain)

1. Concevez un Makefile pour la génération de l'application « `exec` », laquelle est composée de 3 fichiers (file1.c, file2.c et file3.c). Pour la génération de l'application on utilisera le compileur GNU « `gcc` » avec les flags « `-g -Wall -Wextra -O2 -std=c99 -MD` ». Pour rappel, le flag « `-MD` » permet de générer les dépendances. Le Makefile devra également permettre d'effacer les fichiers générés pour une cible donnée. Il est impératif d'utiliser des variables pour spécifier les flags de compilation et les fichiers sources. La génération des codes objets sera fera à l'aide d'une règle.

Makefile :

```
EXEC = exec
CC = gcc
CFLAGS = -g -W -Wall -Wextra -O2 -std=c99 -MD -c
SRCS += file1.c file2.c file3.c
OBJS = $(SRCS:.c=.o)

.c.o:
	$(CC) $(CFLAGS) $(SRCS) $< -o $*.o

all: $(EXEC)

$(EXEC): $(OBJS)
	$(CC) $(LDFLAGS) $^ -o $@

clean:
	rm -Rf $(OBJS) $(EXEC) $(DEP)

.PHONY: all clean
```

2 — 1 — 2 — include $(DEP)

2. Indiquez la fonction des 2 de ces 4 utilitaires suivants :

   a. gcov : utilisé pour tester la couverture d'une application

   b. objdump :

   c. strip :

   d. gprof : (profiling coding) utilisé pour tester la performance des methodes/algorithmes d'une application

2

3. Indiquez en une phrase la méthode pour débugger une application fonctionnant sur une cible à partir d'une machine hôte.

   gdb server? <hôte:port> <application_sans_symboles>

   1 + gdb + gdbserver

   on lance l'appl. sur la cible avec l'utilitaire gdbserver ...

Microprocesseurs 1 & 2: Travail écrit no 4.

5

Problème n° 3 (Vérification)

1. Citez 2 techniques/méthodes permettant de valider des applications logicielles dans les différentes phases de leur développement

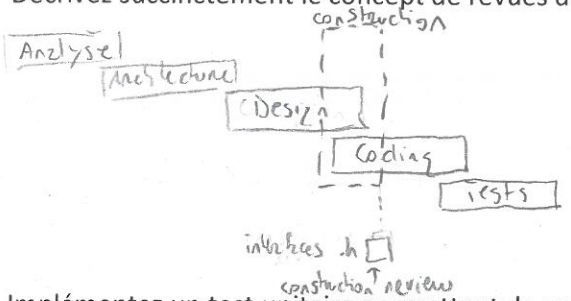
2 - reviews
- tests unitaires
- SCM

2. Décrivez une technique/méthode permettant de garantir qu'un composant logiciel a été correctement et si possible complètement vérifier. 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.

L'exécution de tests périodiques et automatiques

1 gcov : ./exec + testunitaire (test unitaires)

3. Décrivez succinctement le concept de revues de construction



2 Le revues de construction c'est un test qui vient fait vers la fin de la phase de design et le debut de la phase d'implémentation afin de valider le correct fonctionnement de l'application. Ce test est très précis et il permet ainsi d'éviter des efforts supplémentaires en cas de nécessité du code

4. Implémentez un test unitaire permettant de valider/vérifier deux résultats positifs et un résultat négatif pour la fonction « `strrchr ()` » de la librairie standard C (selon description ci-dessous).

```
/** string scanning operation
 *
 * The strrchr() function shall locate the last occurrence of c (converted to a char)
 * in the string pointed to by s. The terminating null byte is considered to be part
 * of the string.
 *
 * @return Upon successful completion, strrchr() shall return a pointer to the byte
 *         or a null pointer if c does not occur in the string.
 */
char *strrchr(const char *s, int c);
```

0

```
const char* string = "0123....9abcdef012";
bool ok = false;
char* p=0;

p = strrchr (string, '0');
ok = p == (string + 16);
p = strrchr (string, 'f');
ok = p ==
:
:
```

Microprocesseurs 1 & 2: Travail écrit no 4.

10

Problème n° 4 (Documentation)

1. Citez 4 outils permettant de simplifier le développement de logiciels et d'améliorer sa qualité

- Source Code Management Tools (GIT, ...)
- Test Tmck Tools (outils qui permettent de générer et tracer les bugs) — pas sûre du nom
- Automatic Build Tools (Makefile, ...)
- Documentation Tools (Doxygen, ...)

3

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

documentation de l'utilisateur

documentation du vérificateur

documentation du développeur

3

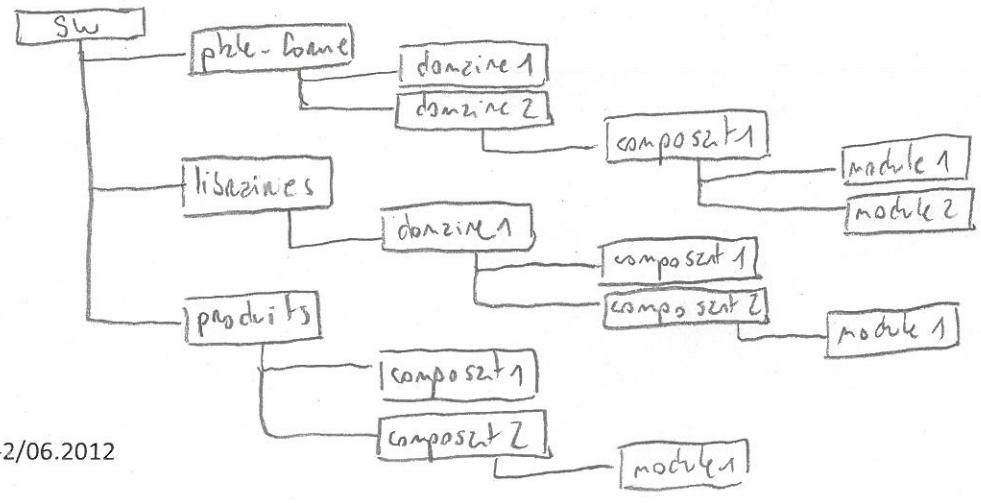
3. Décrivez succinctement l'utilité d'un SCM (Source Code Management Tool) tel que GIT ou SVN

Il permet, à un moment donné, de sauvegarder l'état de l'application (ou seulement les modifications apportés depuis le dernier enregistrement) de façon à pouvoir suivre l'avancement de l'application et surtout de récupérer l'état de l'application à un moment précédent l'état actuel. Il est très utile dans le cas qu'on doit retourner à une version précédente de l'application car elle ne marche plus ou simplement pour observer toutes les modifications apportées, ...

2

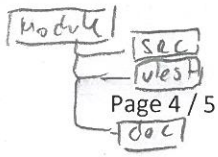
4. Indiquez une manière de structurer le logiciel et sa documentation afin de simplifier son développement et sa maintenance

L'organisation est très importante car elle permet d'améliorer et optimiser le codage et toute une série d'autres avantages. Exemple de structuration:



2

De plus, même au niveau plus bas (module), on peut encore organiser pour simplifier le développement et la maintenance



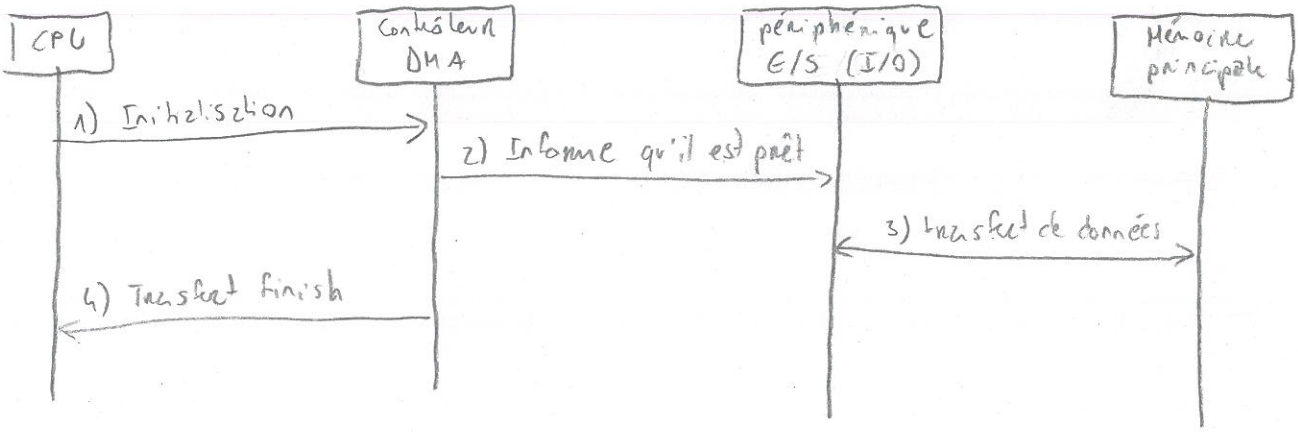
Microprocesseurs 1 & 2: Travail écrit no 4.

Problème n° 5 (Mémoire cache et MMU)

1. Décrivez succinctement le principe d'un DMA.

Afin de décharger le CPU, un contrôleur DMA est utilisé pour gérer les transfert des données entre les périphériques d'E/S (I/O) et la mémoire principale. Pour cela, il doit avoir accès aux signaux de contrôle et aux bus d'adresses et de données. De telle manière le CPU est appelé uniquement lorsqu'il est directement intéressé mais, par contre, les bus seront indisponibles pour d'autres activités lorsqu'un transfert est en train de s'exécuter.

2. Indiquez à l'aide d'un graphique les 4 phases principales d'un transfert DMA



3. Décrivez succinctement la fonction de la mémoire cache et citez les deux principes qui sont à son origine.

Principes : temporel : si à un instant donné, un emplacement de la cache est occupé, il le sera encore, très probablement, peu après.

spatial : si un emplacement de la cache vient d'être écrit, très probablement un emplacement proche de celui-ci sera écrits peu après

4. Citez deux algorithmes de remplacement de ligne dans la mémoire cache

- Random
- Least Recently Used (LRU)
- FIFO
- Least Frequently Used (LFU)

5. Décrivez succinctement les deux algorithmes d'écriture des données dans la mémoire principale cache

- écriture simultanée (write through) : la donnée vient écrite simultanément sur la mémoire cache et sur la mémoire principale

- écriture différée (write-back ou copy-back) : la donnée vient premièrement écrite sur la mémoire cache (doit passer de toute façon par elle). Elle ne vient écrite sur la mémoire principale que lorsque la cache doit libérer l'espace pour une autre donnée. À cet instant la donnée sera transférée vers la mémoire principale