# Sûreté de Fonctionnement
# Master COMASIC
# Contrôle des Connaissances

## Consignes Générales :

- Durée : 2h
- Barème indicatif,
- Il sera tenu compte de la présentation et de la clarté dans la rédaction,
- Seules les réponses précises et justifiées lorsque demandé seront considérées,
- **Vous répondrez aux questions sur le présent sujet, l'espace prévu a été réservé**,
- Mettez vos Noms, Prénoms en haut de chaque recto du sujet.
- Les téléphones portables, tablettes et calculatrices ne sont pas autorisées durant l'examen, et doivent être rangés dans vos sacs.

| Exercice | Barème | Points |
|---|---|---|
| **Attributs de la SdF** | 2 | |
| **Moyen de la SdF** | 2 | |
| **QCM tolérance aux fautes** | 4 | |
| **Arbres de fautes** | 3 | |
| **Coupure Minimale** | 3 | |
| **Signalement et Confinement** | 3 | |
| **Disponibilité et Fiabilité** | 3 | |
| Note | 20 | |

# A. Terminologie et Concepts

**A.1.** (2 points) **Attributs de la SdF**

Rappelez la définition des attributs de la sûreté de fonctionnement ”fiabilité”, ”disponibilité” et ”sécurité-innocuité”. Illustrez chaque définition par une situation affectant négativement l'attribut en question.

**Réponse :**
La fiabilité est la capacité du système à rendre son service tel qu'il est attendu. Si l'on considère un service de base de données renvoyant le crédit associé à des comptes d'impressions, le service est jugé non fiable si le crédit est parfois déclaré nul alors que la base de donnée est censé détenir une valeur strictement positive.
La disponibilité caractérise la capacité du système à rendre tout ou partie du service La disponibilité du service de base de donnée serait affectée si le service ne répond simplement pas aux requêtes de consultation du crédit des comptes d'impression.
La sécurité-innocuité concerne la capacité du système à ne pas engendrer de situations catastrophiques. Pour l'illustrer, le serveur de base de donnée n'est pas approprié. Par exemple la défaillance d'une climatisation en générale dans un véhicule terrestre a une conséquence de l'ordre du confort. En revanche, cette défaillance pour un avion peu avoir des conséquences catastrophiques mettant la vie des passager en danger.

**A.2.** (2 points) **Moyen de la SdF**

Rappelez les définitions de la prévention des fautes et de l'élimination de fautes. Détaillez en quoi ces deux approches diffèrent sur l'exemple du traitement des fautes de programmation entrainant un NullPointerException lors de l'exécution d'un code Java.

**Réponse :**
La prévention de faute consiste à mettre en place une discipline (des règles contraignantes) permettant d'empêcher l'introduction de fautes dans la conception, l'implémentation, ou l'utilisation d'un système.

L'élimination de faute correspond à l'ensemble des moyen disponible pour identifier et corriger la conception ou l'implémentation du système afin que l'activation ne soit plus possible (usuellement en retirant la structure, ou le comportement permettant l'activation de la faute).

Dans le cas d'un programme Java, la prévention des exceptions sur pointeur nul pourrait consister à interdire la déclaration de variables de type objet non initialisée et à forcer l'écriture d'un code de vérification des paramètres de type objet de chaque méthodes. L'élimination de faute en revanche consisterait à utiliser le debugger pour comprendre l'origine de l'apparition de l'exception puis modifier le code source afin d'éviter son apparition.

**A.3.** (4 points) **QCM tolérance aux fautes**

Indiquez la réponse correcte pour chaque question qui suit (1 seule réponse correcte possible, 0 pts par question si plusieurs réponses marquées comme correctes ou si réponse fausse).

**A.3.a.** La distance de Hamming est :

1. un code correcteur d'erreur
2. un opérateur de comparaison de vecteurs de bits
3. un mécanisme de détection d'erreur sur des données stockée

**Réponse :** 2.

**A.3.b.** La réplication passive permet de tolérer

1. des défaillances byzantine
2. des défaillances en valeur
3. des défaillances par crash

**Réponse :** 3.

**A.3.c.** Les recovery blocks sont une implémentation

1. de la réplication active
2. du N version programming
3. d'un recouvrement avant

**Réponse :** 2.

**A.3.d.** Pour tolérer des fautes de développement liées à la non initialisation de pointeurs, il n'est pas correct d'utiliser un mécanisme de :

1. recouvrement avant
2. recouvrement arrière
3. N version programming

**Réponse :** 2.

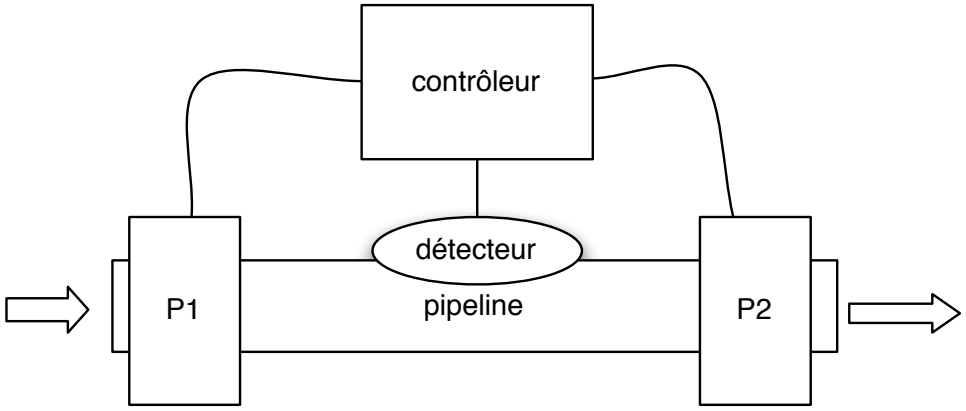


Figure 1: Pipeline : description

# B. Arbres de Fautes

**B.4.** (3 points) **Arbres de fautes**

Nous considérons un système en charge de contrôler la pression d'un liquide sur un segment de pipeline. L'événement redouté étudié est lié à une rupture du pipeline du à une surpression ou une dépression (abaissement non prévu de la pression) dans le pipeline. L'architecture du système est décrite figure 1. Dans le cadre d'un fonctionnement normal, deux pompes assurent le contrôle de la pression dans le pipeline : P1 et P2. Ces pompes lorsqu'elle fonctionnent peuvent être activées ou éteintes. Le pipeline est muni d'un détecteur de pression permettant d'avoir à tout instant une estimation de la pression $P$. Deux seuils de sécurité on été définis, un seuil de pression haute $HP$, et un seuil de pression basse $LP$. Un contrôleur central contrôle la pression et commande l'extinction ou l'allumage des pompes via un signal transmis par des câbles reliant le contrôleur central et chaque pompe ou le détecteur. En l'absence de signaux, les pompes conservent leur mode de fonctionnement. Le comportement attendu est le suivant :

- Tant que le seuil haut est dépassé, la pompe P1 doit s'arrêter et pompe P2 doit continuer de fonctionner.
- Tant que le seuil bas n'est pas dépassé, la pompe P1 doit fonctionner et la pompe P2 doit être éteinte.
- dans le cas où la pression est comprise entre les deux seuils, les deux pompes doivent être allumées (et le sont au démarrage du système).

**B.4.a.** Rappelez quel est l'objectif de la définition d'un arbre de fautes : quel est le contenu des nœuds, et le sens des arcs/connecteurs.

**B.4.b.** En tenant compte de ruptures possibles des câbles, de défaillances des pompes ou du détecteur de pression, proposez un arbre de faute pour ce système.

**Réponse :**

- L'objectif d'un arbre de faute est d'établir le lien logique existant entre un événement dit racine qui correspond soit à une situation de hazard ou une défaillance de haut niveau, et des événements considéré comme étant plus concret correspondant soit à des défaillances du système, soit à des conditions de fonctionnement / usage particulières. Il existe principalement 3 connecteurs : "et", "ou", "conditionnelle". Un oeud rectangulaire est sert à définir une étiquette textuelle sur la sortie d'une des portes logiques "et"/"ou".
- Nous allons simplement définir une formule sur un ensemble d'événement élémentaires. Tout d'abord l'arbre n'a pas besoin, d'être le plus compact possible. En revanche, il convient usuellement de fournir un minimum d'explication sur la structuration de l'ensemble des causes possible sous forme d'une formule logique.

  Notons tout d'abord que la rupture du pipeline se produit dans deux situations : le dépassement de la limite inférieure de pression(LP), ou le dépassement de la limite supérieure de pression (HP). Dans les deux cas, l'absence de réaction du contrôleur est susceptible d'entrainer chaque événement. Cet événement peut être du à une rupture du câble liant le détecteur ou contrôleur ou une défaillance du détecteur lui même. L'autre cause possible est l'absence de réaction coordonnées des pompes suite à la production de la consigne par le contrôleur. Ceci peut se produire si les pompes tombent en panne ou si le câble les reliant au contrôleur est sectionné. Ici la formule dépend du type d'événement (surpression ou sous-pression si l'on suppose que les pompes ont un modèle de défaillance par crash). Supposons que nous ayant les événements DP1, DP2, et DD pour identifier la défaillance de P1, P2 et le détecteur de pression. Nous introduisons aussi RC1, RC2, et RCD pour identifier la rupture des câbles.

  root = ou ( Sous pression, Sur Pression);
  Sur pression = ou (P1 allumée et RC1 et( DP2 ou RC2 et P2 éteinte) ou (DD ou RCD)
  Sous pression = ou (P2 allumée et RC2 et( DP1 ou RC1 et P1 éteinte) ou (DD ou RCD) Une réécriture de l'arbre peut donner : root = ou ( (DD ou RCD), (P1 allumée et RC1 et( DP2 ou RC2 et P2 éteinte), (P2 allumée et RC2 et( DP1 ou RC1 et P1 éteinte)).

**B.5.** (3 points) **Coupure Minimale**

**B.5.a.** Rappelez la définition d'une coupe minimale et illustrez-la sur cet arbre.

**B.5.b.** Expliquez quelle modification peut-on apporter au système pour qu'il n'y ait plus de coupe minimale composée d'un unique événement : "rupture d'un câble". Modifiez l'arbre de fautes en conséquences en détaillant les états intermédiaires si nécessaires.

**Réponse :**
Une coupe minimale est un ensemble de composant tel que la défaillance conjointe de chaque élément de cet ensemble est suffisante à causer l'événement racine de l'arbre, sachant que si n'importe lequel des éléments de l'ensemble était fonctionnel alors le système le serait.
Une coupe minimale sur cet arbre est $\{DD\}$, $\{RC1, DP1\}$ ... Pour éviter qu'il existe une coupe minimale composée de la rupture d'un unique cable, il faudrait dupliquer le câble reliant le détecteur ou contrôleur, ou bien modifier le comportement du contrôleur pour qu'il place le pipeline dans un état sur de fonctionnement (arrêt des deux pompes, ce qui modifierait la spécification de l'attendu).

# C. Confinement de fautes et API

**C.6.** (3 points) **Signalement et Confinement**

Voici la description de l'appel système de déverouiilage d'un mutex :

```
  SYNOPSIS
     #include <pthread.h>

     int  pthread_mutex_unlock(pthread_mutex_t *mutex);

DESCRIPTION
     If the current thread holds the lock on mutex, then the
     pthread_mutex_unlock() function unlocks mutex.
     Calling pthread_mutex_unlock() with a mutex that
     the calling thread does
     not hold will result in undefined behavior.

RETURN VALUES
     If successful, pthread_mutex_unlock() will return zero.
     Otherwise, an
     error number will be returned to indicate
     the error.

ERRORS
     pthread_mutex_unlock() will fail if:
    [EINVAL]           The value specified by mutex is invalid.
    [EPERM]            The current thread does not hold a lock on mutex.
```

**C.6.a.** Rappelez ce qu'est une zone de confinement d'erreur.

**C.6.b.** Un mutex (verrou) permet d'implémenter la notion d'exclusion mutuelle. Cependant, le verrou ne peut tolérer certaines fautes d'interaction (cf description ci-dessus pour l'appel system pthread_mutex_unlock()). Donnez à partir de la description fournie une faute d'interaction clairement identifiée et détectée et une autre qui n'est pas maitrisée. Justifiez votre réponse à chaque fois.

**C.6.c.** Proposez ne démarche reposant sur la prévention de faute permettant d'éviter cette faute d'interaction

**Réponse :**
Une zone de confinement de fautes correspond à une interface délimitant un système ou une partie de ses composant de telle sorte que cette interface soit munie de mécanismes de signalement des erreurs rencontrée, et ayant la capacité de confiner leur propagation.
Une première faute d'interaction serait de "tenter de prendre un mutex un passant comme paramètre un pointeur correspondant à un mutex non initialisé". Il en résulte usuellement

soit une erreur EINVAL, soit un segmentation fault. Une seconde faute consiste à relâcher un mutex que vous n'avez pas pris. Dans ce cas l'erreur est détectée localement mais pas notifiée globalement. Ainsi, si le traitement d'erreur local est incorrect, la faute peut se propager aux autres thread partageant le mutex.

# D. Analyse quantitative

**D.7.** (3 points) **Disponibilité et Fiabilité**

Nous supposons que l'on dispose de modules de calcul ayant une fiabilité par sollicitation de $r$ et taux de crash par heure de $c$ (entrainant un défaut de disponibilité). L'architecture suivante a été retenue pour assurer une bonne fiabilité et disponibilité : 5 répliques composent le système. Pour être disponible au moins trois répliques doivent être en état de s'exécuter (en fait on en fait exécuter exactement trois quelque soit le nombre disponible au delà). En permanence, toutes les répliques sont sous tension et peuvent être victime d'une défaillance par crash mais seules trois réalisent les calculs. Pour que la valeur produite soit correcte, deux des trois répliques actives doivent avoir produit un résultat correct.

**D.7.a.** Donnez l'expression définissant la fiabilité de ce système

**D.7.b.** Donnez la formule logique à partir des états de chaque réplique qui est vraie lorsque le système est disponible

**D.7.c.** Donnez l'expression quantitative de la disponibilité de ce système par heure de fonctionnement.

**Réponse :**

- La formule correspondant à la fiabilité du système correspond à la somme des probabilités de 3 composants correct + exactement 2 composants corrects parmi 3 : $R_{sys} = r^3 + 3(1-r)r^2$.
- Notons $x_1, ...x_5$ les états des répliques (vrai si fonctionnels) :

$$\begin{aligned} D(x_1, \cdot, x_5) &= (x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge x_4) \\ &\vee (x_1 \wedge x_2 \wedge x_5) \vee (x_1 \wedge x_3 \wedge x_4) \\ &\vee (x_1 \wedge x_3 \wedge x_5) \vee (x_1 \wedge x_4 \wedge x_5) \\ &\vee (x_2 \wedge x_3 \wedge x_4) \vee (x_2 \wedge x_3 \wedge x_5) \\ &\vee (x_2 \wedge x_4 \wedge x_5) \vee (x_3 \wedge x_4 \wedge x_5) \end{aligned}$$

- $D(c) = (1-c)^5 + 4 * (1-c)^4 * c + 10 * (1-c)^3 * c^2$