ArcSys1 : Systèmes et Réseaux

Examen final du 19 décembre 2017 (2 h)

Documents autorisés : un A4 recto (manuscrit de votre main) et les pense-bêtes du cours (C et Unix).
La correction tiendra compte de la qualité de l'argumentaire et de la présentation.
Un code illisible et/ou incompréhensible est un code faux. Le barème indiqué est approximatif.

★ **Exercice 1 : Chaînes de caractères et fichiers (4 pts).** *Le ROT13 ("rotate by 13 places") (une variante du chiffre de César) est un algorithme très simple de chiffrement de texte. Comme son nom l'indique, il s'agit d'un décalage de 13 caractères de chaque lettre du texte à chiffrer. [...] L'avantage de ROT13, c'est le fait que le décalage soit de 13. L'alphabet comporte 26 lettres, et si on applique deux fois de suite le chiffrement, on obtient comme résultat le texte en clair. Pour cela on doit considérer l'alphabet comme circulaire, c'est-à-dire qu'après la lettre Z on a la lettre A, ce qui permet de grandement simplifier son usage et sa programmation puisque c'est la même procédure qui est utilisée pour le chiffrement et le déchiffrement.* (D'après Wikipédia).

Cara. non-chiffré	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Caractère chiffré	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M

▷ **Question 1 :** Écrivez une fonction `void rot13(char *chaine)` appliquant l'algorithme ROT13 à la chaîne passée en argument. Vous avez le droit de modifiez la chaîne en place ou de la copier avant de la modifier.

▷ **Question 2 :** Écrivez un programme complet prenant une chaîne de caractères à analyser en argument de la ligne de commande, de la manière suivante :

```
$ ./rot13 "Chiffre moi"
Puvsser zbv
```

▷ **Question 3 :** Modifiez votre programme afin qu'il lise le texte à analyser depuis un fichier dont le nom est passé en argument de la ligne de commandes.

★ **Exercice 2 : Lecture de code (4pts).**

Indiquez l'affichage des programmes C suivants, qui compilent et s'exécutent sans erreur.

Programme A

```c
#include <stdio.h>
void f1 (int a, int b) {
  a += b;
}
void f2 (int* a, int* b) {
  *a += *b;
}
void f3 (int *a) {
  static int b = 0;
  b++;
  *a += b;
}
int main () {
  int a = 5, b = 7;
  f1( a, b );
  printf("a=%d, b=%d\n",a,b);
  int c = 9, d = 11;
  f2( &c, &d );
  printf("c=%d, d=%d\n",c,d);
  int e = 15;
  f3(&e);
  printf ("e=%d\n", e);
  f3(&e);
  printf ("e=%d\n", e);
  return 0;
}
```

Programme B

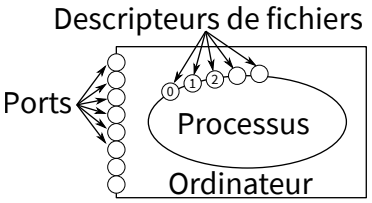
```c
#include <stdio.h>

int main() {
  int tab[4] = {3, 2, 1, 0};
  int* p = tab;
  while (*p) {
    (*p)++;
    p++;
  }
  printf("%d %d %d %d\n",
         tab[0], tab[1],
         tab[2], tab[3]);
  return 0;
}
```

Programme C

```c
#include <stdio.h>

void g(int* a, int *b) {
  int t = *a;
  *a = *b;
  *b = t;
}

int main() {
  int tab[4] = {42, 72};
  int* u = tab;
  int* v = tab + 1;
  g(u, v);

  printf("%d %d\n",
         tab[0], tab[1]);
  return 0;
}
```

★ **Exercice 3 : Sockets BSD (4pts).**

On rappelle le formalisme vu en cours pour représenter des processus en cours d'exécution, repris ci-contre. Les machines sont représentées par des rectangles, avec des petits ronds sur le coté pour représenter les ports de la machine. Au sein des machines, chaque processus est représenté par une ellipse, et des petits ronds à la bordure de cette ellipse représente les différents descripteurs de fichier du processus. Les trois premiers sont habituellement `stdin`, `stdout`, `stderr`.



▷ **Question 1 :** Dessinez en respectant ce formalisme les différentes étapes de l'établissement d'une connexion TCP entre deux processus distants. Vous donnerez les appels systèmes utilisés par chaque partie. Vous dessinerez l'état des choses après chaque appel système, en ajoutant les explications adéquates. En revanche, il ne vous est pas demandé de donner le code exact réalisant ces appels systèmes.

★ **Exercice 4 : Contrôle de congestion (3pts).**

On considère une connexion TCP établie, avec `SS_thres=32`. À t_0, $W = 1$. Dix émissions de paquets ont lieu aux temps t_0 à t_9, d'abord en régime *slow start* puis en régime *congestion avoidance*. À t_{11}, l'émetteur reçoit un triple ACK. À t_{20}, il constate un timeout de ses envois.

▷ **Question 1 :** Faites un graphique de la taille de la fenêtre utilisée coté émetteur dans ce scénario en fonction du temps, sur la période $[t_0; t_{25}]$. Vous préciserez les périodes de SS et celles de CA.

★ **Exercice 5 : Routage IP (3pts).**

Un ami vous appelle à l'aide pour réparer son réseau personnel. Il possède deux ordinateurs (1 fixe et 1 portable), un réfrigérateur connecté, et un routeur qui est connecté à Internet. Les terminaux (fixe, portable et réfrigérateur) sont reliés à un switch qui est lui même relié au routeur. Les adresses IP sont configurées manuellement comme suit :

Machine	**Adresse IP**	**Masque**	**Passerelle**
Ordinateur fixe	192.168.1.5	255.255.255.224	192.168.1.1
Ordinateur portable	192.168.1.33	255.255.255.0	192.168.1.1
Réfrigérateur	10.28.2.3	255.0.0.0	10.0.0.1 (Internet)
Routeur	192.168.1.1	255.255.255.0	10.0.0.1 (Internet)

▷ **Question 1 :** Pourquoi son réfrigérateur n'a pas accès à Internet bien que le câble soit correctement branché ? Comment corriger ?

▷ **Question 2 :** Pourquoi est-il plus lent d'envoyer un paquet du fixe au portable que du portable au fixe ? Comment corriger ?

▷ **Question 3 :** Qu'est ce que DHCP, et en quoi ce protocole peut-il aider pour les problèmes rencontrés sur cette infrastructure ?

★ **Exercice 6 : Questions de cours (2pts).**

▷ **Question 1 :** Discutez les avantages comparés de gdb et valgrind (15 lignes max).

▷ **Question 2 :** Discutez les différences et ressemblances entre les tableaux et les pointeurs en C (15 lignes max).