Université de Limoges | FACULTÉ DES SCIENCES ET TECHNIQUES

Master 1ère année

Réseaux & Système

Janvier 2014

Durée : 2h — Documents autorisés : le support « Mon'tit Python » uniquement

Unix & Gestion des processus — (5 points)

1 – 5pts Questions sur le fonctionnement du système Unix :

a. Qu'est-ce que le « *multi-tâche préemptif* » ?
Comment est garantie l'équité de traitement des différents processus s'exécutant dans le système ?

b. Comment est-il possible de passer du « *mode utilisateur* » au « *mode noyau* » ?
Vous citerez différentes possibilités.

c. Expliquez comment fonctionne le « *fork* » ?
Comment est-il possible d'en tirer parti en programmation ?

d. Quelles sont les différences entre un processus et des *threads* ?
Comment est gérer l'ordonnancement dans les deux cas ?

e. Expliquez pourquoi dans un système Unix, deux processus s'exécutant en multi-tâche ne peuvent accéder, chacun, à l'espace mémoire de l'autre.
Vous décrirez les procédés physiques intervenant dans cette protection.

Segmentation & Pagination — (7 points)

2 – 4pts Soit un processeur utilisant des « *mots* » de 64 bits et utilisant un adressage sur 64 bits.
On veut mettre en place de la « *mémoire vituelle* » avec des pages d'une taille de 65536 octets.

a. Expliquez comment va fonctionner le mécanisme de pagination ?

b. Sur un exemple d'adresse, illustrez comment on passe d'une adresse mémoire contenue dans une page à une adresse mémoire contenue dans la page suivante.

c. Qu'est-ce qu'un « *défaut de page* » ?

d. Est-il possible d'utiliser une page pour un segment quelconque ?
Peut-on empêcher les utilisations frauduleuses d'une page ?

3 – 3pts Soit le morceau de programme suivant :

```
1  #include <stdlib.h>
2  #include <malloc.h>
3
4  int *initialiser_tableau(int coeff)
5  { int i;
6    int *tab = (int *) malloc(sizeof(int)*10);
7
8    for (i=0; i < 10; i++)
9    { tab[i] = i * a; a = a + i; }
10
11   return tab;
12 }
13
14 int main()
15 {
16   int *donnees = initialiser_tableau(10);
17   ...
18 }
```

a. Donnez l'état de la pile lors de l'appel de la fonction en ligne 17 :

Pile/Stack	Tas/Heap
...	...

b. Donnez l'état de la pile lors du retour de la fonction en ligne 18 :

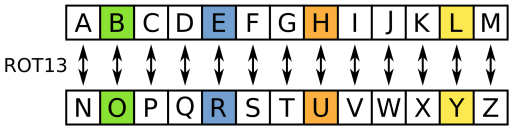
Pile/Stack	Tas/Heap
...	...

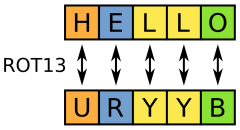
c. Transformez le programme pour faire un programme correct « allocation dynamique ».

Programmation Python & Réseau — 8 points

4 – 4pts La méthode de « dissimulation » appelée « **ROT13** » est une simple méthode de **substitution** dont le but est de rendre illisible une partie d'un texte lors de la lecture d'un document : ne pas révéler l'intrigue d'un film sans que l'utilisateur le désire, exprimer des idées à l'abri de « robots » parcourant les données de manière automatique, abuser les « *noobs* », *etc.*

En utilisant uniquement les 26 lettres de l'alphabet, on peut faire une seule opération permettant de dissimuler et de rendre visible en décalant de 13 systématiquement le rang de chaque caractère :

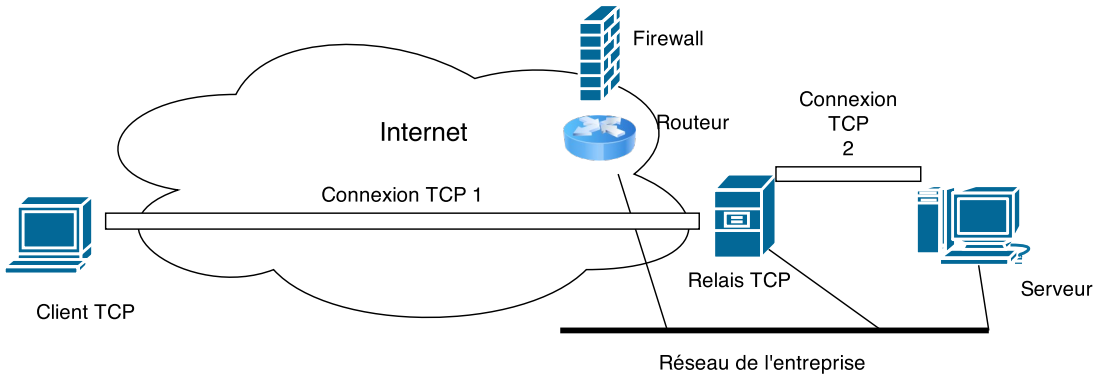




a. Écrivez une fonction Python permettant de réaliser une opération « `ROT13` » sur le contenu d'une chaîne de caractères ;

b. Écrivez un programme Python utilisant la fonction précédente en saisissant une chaîne de caractères de la part de l'utilisateur et en gérant la présence des seuls caractères alphabétiques.
Vous choisirez soit d'ignorer les caractères non alphabétiques, soit de les filtrer.

5 – 4pts Le réseau d'une entreprise a été sécurisé à l'aide d'un *firewall filtrant* de telle manière que la machine « Serveur » connectée à l'intérieur du « Réseau de l'entreprise » soit inaccessible directement depuis Internet.



Une machine « Relais TCP » a été installée de manière à relayer les connexions depuis Internet vers la machine « Serveur » :

- ▷ elle dispose de l'adresse IP `193.50.13.56` ;
- ▷ elle est autorisée à recevoir des connexions TCP depuis Internet vers le port `5000` ;
- ▷ elle réalise le travail suivant :
  - ◇ la machine « Client TCP » établie une connexion vers la machine « Relais TCP » : « Connexion TCP 1 » ;
  - ◇ la machine « Relais TCP » établie une connexion vers le « Serveur » : « Connexion TCP 2 » ;
  - ◇ les données de « Client TCP » sont relayées vers « Serveur » jusqu'à la fin de la connexion « Connexion TCP 1 » ;
  - ◇ la connexion « Connexion TCP 2 » est alors fermée.

La configuration du réseau de l'entreprise est la suivante :

- ⋆ la machine « Serveur » possède l'adresse IP `193.50.13.17` ;
- ⋆ le réseau de l'entreprise est configuré en `193.50.13.0/24` ;
- ⋆ le routeur/firewall possède l'adresse IP `193.50.13.254`.

Questions :

On vous demande d'écrire le programme réseau réalisant le travail de « Relais TCP » :

a. Quel TSAP doit utiliser « Client TCP » pour se connecter ?

b. Dans quel mode doit fonctionner le programme pour dialoguer avec « Client TCP » ?
Indiquez quelles opérations de la programmation Socket permettent de le mettre en œuvre et les paramètres à utiliser.

c. Lorsque le programme réseau doit établir la seconde connexion TCP, dans quel mode doit-il fonctionner ?
Indiquez quelles opérations de la programmation Socket permettent de le mettre en œuvre et les paramètres à utiliser.

d. Écrivez le corps du programme réseau en Python réalisant le relais des données en provenance de « Client TCP » vers « Serveur ». Quand et comment finira-t-il son travail ?
Vous utiliserez ce que vous avez écrit précédemment en spécifiant le nom des variables nécessaires.

e. Si on veut pouvoir faire le relais des données de « Client TCP » → « Serveur » simultanément avec le relais des données de « Serveur » → « Client TCP », comment peut-on procéder et avec quelle(s) fonctionnalité(s) système ?