Université de Nice-Sophia Antipolis L2 Info
Examen 2009–2010

Systèmes d'exploitation

Nom :_____

Prénom :_____

Durée : 2h

Nombre d'intercalaires :

Note :

NB : Vos réponses aux questions doivent être brèves et synthétiques. Il est inutile de faire de longs discours.

► 1. (3pts) Supposons que le fichier `toto.txt` contienne le texte suivant :

```
0123456789Ceci est le contenu du fichier.
```

Expliquez quel sera l'affichage produit par le programme suivant :

```
#!/bin/env python
import sys,os

fd1 = os.open("./toto.txt",os.O_RDONLY)
buf = os.read(fd1,8)
pid = os.fork()
if pid == 0:
   for i in range(3):
       fd2 = os.open("./toto.txt",os.O_RDWR|os.O_APPEND)
       os.write(fd2,"truc")
       os.close(fd2)
   print buf
   sys.exit(0)
else:
   os.wait()
   os.execlp("cat","cat","./toto.txt")
```

► 2. (7pts) Problème (les 2 parties sont indépendentes). On souhaite écrire un script appelé `monmail.py` qui implémente un client mail simple, qui permet d'envoyer un message à un unique destinataire, mais à condition de préciser l'expéditeur (on verra pourquoi plus loin). Un exemple d'utilisation est donné ci-après (`shell>` reprśente le prompt du shell unix, et `^D` la saisie d'une fin de fichier par la combinaison de touches Control+D) :

```
shell> python monmail.py expediteur@domaine-exp destinataire@domaine
Subject: un test de mail

contenu du mail (ligne1)
contenu du mail (ligne2)
contenu du mail (ligne3)
^D
```

Ce client devra donc se connecter à un serveur de mail distant et dialoguer avec lui dans le protocole SMTP, l'équivalent pour le mail du protocole HTTP pour le web. Comme HTTP, utilise une connexion TCP, mais sur le port 25 au lieu du port 80 pour HTTP.

Partie 1 (3pts) Avant de regarder de plus près SMTP, voici quelques petites questions pour se préparer à résoudre le problème.

- En python, quelles instructions permettent à un client d'établir une connexion TCP (donc en mode connecté), sur le port 25 d'une machine appelée machine.unice.fr?

- Donnez le code de la procédure python `envoyer_req(sockfd,msg)` qui envoie le message contenu dans le paramètre `msg` sur un socket contenu dans le paramètre `sockfd` déjà connecté avec un serveur en mode `SOCK_STREAM` (TCP).

- Donnez le code de la fonction Python `code,msg = recevoir_rep(sockfd)` qui reçoit une réponse du serveur sur un socket TCP déjà connecté. On supposera que cette réponse est **toujours** formée d'une seule ligne, commençant par un nombre à 3 chiffres (code de retour) et d'un texte de longueur variable (le message), comme dans les exemples suivants:

```
250 recipient <Olivier.Dalle@sophia.inria.fr> ok
250 ok:  Message 62836045 accepted
500 #5.5.1 command not recognized
```

Votre fonction doit retourner un tuple formé de deux éléments : lu code de retour et le texte du message, extraits du message reçu par le socket.

Partie 2 (4pts) Voici ci-après une description textuelle simplifiée du protocole SMTP vu du coté client. Etudiez-la attentivement avant de répondre aux questions qui suivent.

- *Lors de l'ouverture de la session SMTP, la première commande à envoyer est la commande EHLO suivie d'un espace et du nom de domaine de votre machine (afin de dire "bonjour je suis telle machine"), puis valider par entrée.*
- *La seconde commande est "MAIL FROM:" suivie de l'adresse email de l'expéditeur. Si la commande est acceptée le serveur renvoie le message "250 OK"*
- *La commande suivante est "RCPT TO:" suivie de l'adresse email du destinataire. Si la commande est acceptée le serveur renvoie le message "250 OK"*
- *La commande DATA est la troisième étape de l'envoi. Elle annonce le début du corps du message. Si la commande est acceptée le serveur renvoie un message intermédiaire numéroté 354 indiquant que l'envoi du corps du mail peut commencer et considère l'ensemble des lignes suivantes jusqu'à la fin du message repéré par une ligne contenant uniquement un point. Le corps du mail contient éventuellement certains des en-têtes suivants : Date, Subject, Cc, Bcc, From.*

  Si la commande est acceptée le serveur renvoie le message "250 OK" (ou 354 si data).

Voici un exemple simplifié d'échange entre un client etiqueté `C` et un serveur etiqueté `S` :

```
C: ehlo inria.fr
S: 250-mail4-relais-sop.national.inria.fr
C: bla-bla?
S: 500 #5.5.1 command not recognized
C: RCPT TO: Olivier.Dalle@sophia.inria.fr
S: 503 #5.5.1 MAIL first
C: MAIL FROM: Olivier.Dalle@sophia.inria.fr
S: 250 sender <Olivier.Dalle@sophia.inria.fr> ok
C: RCPT TO: Olivier.Dalle@sophia.inria.fr
S: 250 recipient <Olivier.Dalle@sophia.inria.fr> ok
C: DATA
S: 354 go ahead
C: Subject: un sujet rigolo!
C:
C: sjdkdhks
C: ldhlha
C: <CRLF>.<CRLF>
S: 250 ok:  Message 62836101 accepted
C: quit
```

(Notez dans cet exemple que le serveur exige que la requête MAIL FROM soit envoyée avant la requête RCPT TO).

En supposant que les fonctions `envoyer_req(sockfd,msg)` et `code,msg = recevoir_rep(sockfd)` décrites dans la première partie sont déjà écrites, et qu'une troisième fonction

```
sockfd = etablit_connexion(destinataire)
```

qui retourne un socket TCP connecté avec le serveur SMTP du destinatire existe, donnez le reste du code du programme `monmail.py`.

► 3. (6pts) Faites un schéma pour représenter la généalogie des processus obtenue à l'aide du programme suivant (attention à l'indentation!) et deduisez-en le nombre de processus différents qui exécutent la dernière instruction du programme (`pass`) :

```
import sys,os
for i in range(2):
    pid = os.fork()
if pid == 0:
    pid = os.fork()
    if pid == 0:
        sys.exit(0)
    else:
        time.sleep(2)
        pid = os.fork()
while True:
    pass
```

Expliquez pourquoi ce programme peut génŕer des processus zombies et donnez une version corrigée de ce programme, aboutissant à la même généalogie, mais en éliminant proprement les processus zombies (attention, vous ne devez pas supprimer d'instruction, seulement ajouter celles qui suppriment les zombies!).

▶ 4. (4pts) Recopie de fichier caractère par caractère. Donnez le code du programme Python `cp.py` qui recopie caractère par caractère le fichier dont le nom est donné en premier paramètre dans le fichier dont le nom est donné en deuxième paramètre. Par exemple, la commande suivante :

```
cp.py toto.txt ../titi.txt
```

recopie le contenu du fichier `toto.txt` dans le fichier `../titi.txt`.

► 5. (4pts) Pour chacune des commandes shell suivantes, donnez un programme Python qui réalise cette commande à l'aide des primitives vues en cours, et sans générer de zombies :

- `ls -al | more`

- `cat < toto | wc -l > titi`