UE Systèmes et Réseaux

Examen de Systèmes

Durée : 1 h 30

Tous documents autorisés. Appareils électroniques interdits.

Si vous avez un doute sur la syntaxe d'une commande où sur le caractère qui remplit un certain rôle dans le langage de commande, utilisez votre propre syntaxe, ou un autre caractère, mais ajoutez un commentaire explicatif.

SUJET (3 pages)

1 Questions de Cours (4 points, 15 à 20 minutes)

1.1 – Multi-tâche

a) Comment le système procède-t-il pour partager le processeur entre les processus présents dans le système ? Expliquez brièvement le principe de fonctionnement de l'algorithme du tourniquet.

b) Quels sont les inconvénients et les avantages d'un quantum court ?

1.2 – Processus vs thread

Expliquez brièvement ce qu'est un processus et quelle différence il y a entre un processus et un thread.

2 Question au sujet de Travaux Pratiques (3 points)

Programmation client/serveur en Java :

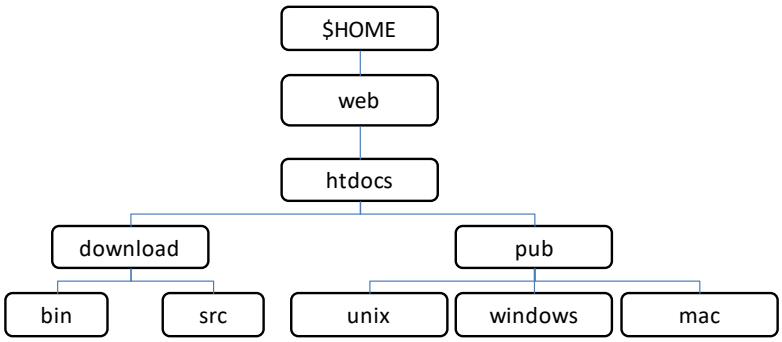
2.1 – Expliquez ce que fait l'instruction suivante extraite du programme client générique programmé en TP :

```
sClient = new Socket(host.getText(), Integer.parseInt(port.getText()));
```

2.2 – Citez les erreurs qui peuvent survenir durant cette opération et les exceptions Java correspondantes à prendre en compte dans le programme.

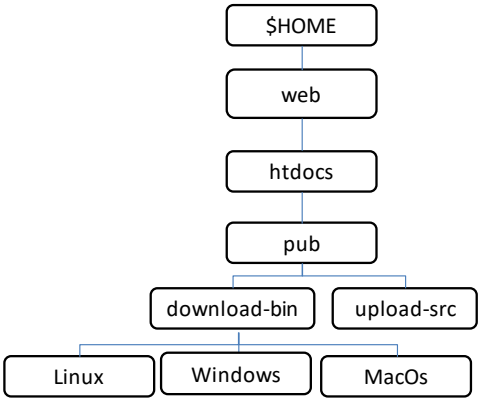
3 Shell Linux (6 points, 30 minutes)

Soit la hiérarchie de répertoires suivante, à laquelle vous avez un accès total. Les répertoires sont supposés non vides.



1) Vous êtes actuellement positionné sur le répertoire `unix` et $HOME a pour valeur `/home/miashs` Votre login est `miashs`. Quels sont respectivement les chemins relatif et absolu du répertoire `mac` ?

2) Donnez les commandes shell nécessaires pour modifier cette hiérarchie de répertoires et aboutir au résultat ci-dessous, sachant que les contenus des anciens répertoires `bin` et `src` doivent être transférés dans les nouveaux répertoires `download-bin` et `upload-src`, respectivement. Le répertoire `download-bin` contiendra également les fichiers de l'ancien répertoire `download`. Les répertoires `unix`, `windows`, `mac` sont renommés respectivement `Linux`, `Windows` et `MacOs` :



3) Ecrire le script shell contenant les commandes nécessaires pour déplacer tous les fichiers éventuellement présents dans le répertoire `pub` et dont le nom se termine par `.java` ou `.c` dans le répertoire `upload-src`, et déplacer tous les fichiers restants du répertoire `pub` dans le répertoire de nom `save` que vous aurez créé au préalable dans votre répertoire personnel s'il n'existe pas déjà, attention, un fichier de nom save peut empêcher la création du répertoire.

4) Donnez les commandes shell linux permettant d'accorder les permissions suivantes :
   - accès complet au répertoire `pub` et à tous ses sous-répertoires, pour vous-même, et accès en lecture et en exécution pour tous les autres utilisateurs (la commande `chmod` accepte l'option `-R` (récursif));
   - accès en écriture au répertoire `upload-src` pour tous les utilisateurs de votre groupe ;
   - aucun accès au répertoire `Linux` pour les utilisateurs autres que vous et les utilisateurs de votre groupe.

4 Processus Unix (7 points, 30 minutes)

a) Expliquez ce que fait l'instruction C : `wait(NULL)`

b) Soit le programme C / Unix suivant :

```
#include <stdio.h>
#include <stdlib.h> /* Pour exit */
#include <unistd.h> /* Pour fork, getpid, getppid, sleep */
#include <sys/types.h> /* Pour pid_t (fork, getpid, getppid) */
#include <sys/wait.h>  /* Pour wait */
int main(void)
{
  pid_t id1;
  pid_t id2;
  id1 = fork();
  printf("Bonjour!\n");
  if (id1 == 0)
  {
     sleep(2);
     printf("Je suis %i, ca va toi ?\n", getpid());
     id2 = fork();
     if (id2 == 0)
     {
         sleep(3);
         printf("Pas mal, je suis %i\n",getpid());
     }
     else {
         wait(NULL);
         printf("Je suis %i, c'est fini ?\n", getpid());
     }
  }
  else {
     sleep(1);
     printf("Je suis %i, et toi ?\n", getpid());
     wait(NULL) ;
     printf("Je suis %i, c'est fini.\n", getpid());
  }

  printf("Je suis %i, au revoir.\n",getpid());
  return EXIT_SUCCESS;
}
```

On suppose que fork() n'échoue jamais (ne renvoie jamais -1).

a) Combien de processus sont créés par ce programme ?

b) Faire la trace d'exécution de ce programme en mettant en évidence ce qu'exécute chaque processus et écrire ce que le programme va afficher sur l'écran, **dans le bon ordre**, en justifiant votre réponse. Pour cela, on suppose que le processus qui exécute ce programme (le père) porte le numéro 1000 et que les processus créés prendront les numéros suivants (1001, 1002, …).

c) Y a-t-il plusieurs traces d'exécution différentes possibles ? Pourquoi ?