

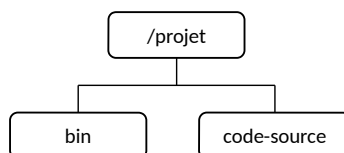
Si vous avez un doute sur la syntaxe d'une commande où sur le caractère qui remplit un certain rôle dans le langage de commande, utilisez votre propre syntaxe, ou un autre caractère, mais ajoutez un commentaire explicatif. Les exercices peuvent être effectués dans n'importe quel ordre.

1- Question de cours (3 points)

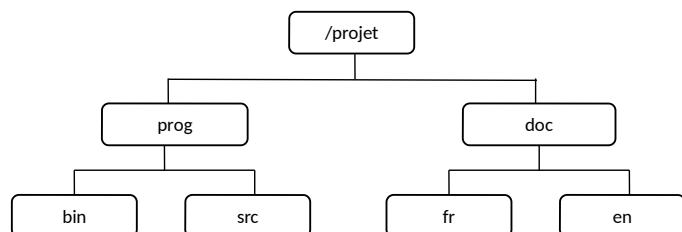
1. Expliquez brièvement ce qu'est la multiprogrammation.
2. Quelles sont les différents états dans lesquels peuvent se trouver des processus ? Expliquez ces états et les transitions entre état possibles.
3. Expliquez la différence entre les stratégies d'ordonnancement préemptive et non-préemptive.

2- Shell UNIX (8 points)

Le répertoire `/projet` contient actuellement l'arborescence suivante :



1. Donnez la suite de commandes shell UNIX permettant d'obtenir la hiérarchie suivante :



Nb : les répertoires `bin` et `code-source` ne sont pas forcément vides, et le nom du répertoire `code-source` doit être changé en `src`.

2. Dans votre répertoire personnel vous disposez d'un répertoire nommé `bazar` contenant un certain nombre de fichiers à transférer dans l'arborescence que vous venez de modifier :
3. Déplacez le fichier `README` du répertoire `bazar` de votre répertoire personnel dans le répertoire `/projet` et déplacez tous les fichiers du répertoire `bazar` se terminant par `.txt` dans le répertoire `doc/fr`
4. Déplacez dans le répertoire `src` tous les fichiers du répertoire `bazar` de votre répertoire personnel dont le nom se termine par `.java`
5. Déplacez dans le répertoire `bin` tous les fichiers du répertoire `bazar` de votre répertoire personnel dont le nom commence par `tableau`, et tous les fichiers de suffixe `.class`
6. Détruire le répertoire `bazar` de votre répertoire personnel ainsi que tout son contenu.
7. On suppose que l'on dispose d'un programme nommé `translate` capable de traduire un texte du français en anglais. `translate` accepte des données sur l'entrée standard, fournit la traduction sur la sortie standard et des indications sur les mots ou phrases potentiellement mal traduits sur la sortie d'erreur. Donnez la commande UNIX permettant de traduire le texte du fichier `doc-fr.txt` (situé dans le répertoire `doc/fr`) dans le fichier `doc-en.txt` (répertoire `doc/en`), et de placer l'état des erreurs dans le fichier `trad-err.txt` dans le répertoire `/tmp`.
8. On veut appliquer la politique de droits d'accès suivante : vous conservez tous les droits sur tous les fichiers du répertoire `/projet` et de ses sous-répertoires. Les membres de votre groupe de travail ont droit de lecture et d'exécution sur tous ces fichiers, mais pas le droit d'écriture. Les autres utilisateurs ont droit de lecture uniquement sur les fichiers des répertoires `fr` et `en`, droit d'exécution uniquement sur les fichiers du répertoire `bin` et aucun accès au répertoire `src`. Donnez la séquence de commandes UNIX permettant de mettre cette politique en œuvre.

Indication : la commande `chmod` peut s'appliquer récursivement sur une arborescence de répertoires avec l'option `-R`.

En supposant que vous vous trouvez dans le répertoire `/projet`, donnez le chemin relatif et le chemin absolu du répertoire `bin`.

3- Threads et synchronisation (6 points)

```
1. class Devinette {
2.
3.     private Object[] tab;
4.     private int t;
5.     private int q;
6.     private volatile int nb;
7.
8.     public Devinette(int n) {
9.         tab = new Object[n];
10.        t=0; q=0; nb=0;
11.    }
12.
13.    public synchronized void m1(Object o) {
14.        while (nb>=tab.length) {
15.            // ne fait rien
16.        }
17.        tab[q]=o;
18.        q=(q+1)%tab.length;
19.        nb+=1;
20.    }
21.
22.    public synchronized Object m2() {
23.        while (nb==0) {
24.            // ne fait rien
25.        }
26.        Object res = tab[t];
27.        t=(t+1)%tab.length;
28.        nb-=1;
29.        return res;
30.    }
31.}
32.
33. public class Exam2022 {
34.
35.     public static void main(String[] args) {
36.         Devinette d = new Devinette(5);
37.         Thread t1 = new Thread() {
38.             public void run() {
39.                 for (int i=0 ; i<1000 ; i++) {
40.                     d.m1(new Object());
41.                 }
42.             }
43.         };
44.
45.         Thread t2 = new Thread() {
46.             public void run() {
47.                 for (int i=0 ; i<1000 ; i++) {
48.                     Object o = d.m2();
49.                 }
50.             }
51.         };
52.         t1.start();
53.         t2.start();
54.     }
55.
56. }
```

1. Quel problème classique de synchronisation simule ce programme ?
2. A quoi servent les mots clé `synchronized` utilisés ici. Quel aurait été le problème si ils n'avaient pas été utilisés ?
3. Dans quels cas ce programme ne se termine pas ? Donnez un exemple.
4. Proposez une solution qui permet de garantir que le programme se termine tout en assurant que tous les objets créés par `t1` soit récupérés par `t2`.

4- Processus Unix (3 points)

On considère le programme suivant :

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
int main(void){
    pid_t id;
    printf("Bonjour\n");
    id=fork();
    printf("alors\n");
    if (id == 0) {
        id=fork();
        if (id == 0) { sleep(3); printf("effectivement\n"); }
        else {
            printf("bien entendu\n");
            wait(NULL);
            printf("c'etait a prévoir\n");
        }
    }
    else {
        sleep(5); printf("oui en effet\n");
        wait(NULL);
        printf("vraiment etonnant\n");
    }
}
```

1. Donnez la trace d'exécution, c'est-à-dire le texte affiché à l'écran lors de l'exécution du programme, justifiez votre réponse.
2. Combien de processus sont créés par ce programme ?
3. Il y a plusieurs traces d'exécution possibles, pourquoi ? Donnez une seconde trace d'exécution.