Épreuve disciplinaire appliquée

Préambule : cette épreuve est constituée de deux parties A et B indépendantes. Les réponses aux questions doivent être précises et rédigées avec soin.

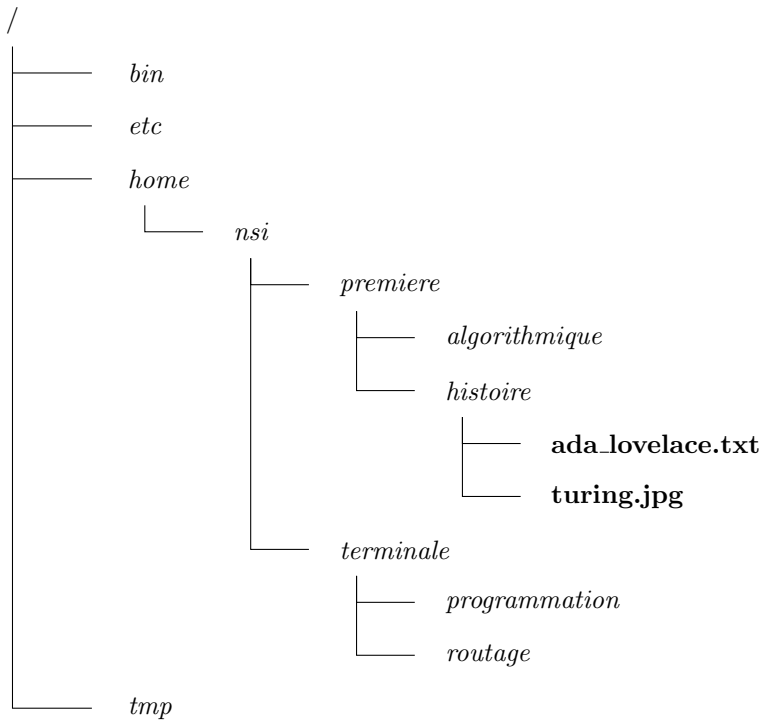
Certaines questions demandent des productions d'enseignement à destination des élèves (cours, exercices, projets, activités sur machines, activités débranchées, etc.). Ces questions devront être traitées avec la plus grande attention.

Partie A - Systèmes d'exploitation et processus

Dans cette partie, on se propose d'étudier les systèmes d'exploitation et les processus, notions qui font partie des programmes de NSI de première et de terminale. Elle est composée de deux sous-parties, l'une sur les systèmes d'exploitation et l'autre sur les processus et leur ordonnancement.

1 Systèmes d'exploitation

1. Donner la définition d'un système d'exploitation et préciser trois de ses principales fonctionnalités.
2. Donner un exemple de système d'exploitation propriétaire et un exemple de système d'exploitation libre.
3. Donner deux caractéristiques principales des systèmes Unix récents (compatibles avec le standard POSIX).
4. Dans un système Unix, un utilisateur se voit attribuer un UID et un GID. Expliquer à quoi cela correspond.
5. Préciser à quoi correspond le super-utilisateur d'un système d'exploitation de type Unix. Donner son UID et son GID utilisés par convention dans les systèmes Unix.
6. Dans un système d'exploitation de type Unix, on considère l'arborescence des fichiers suivante dans laquelle les noms de répertoires sont en italique et ceux des fichiers sont en gras :



On se place dans un terminal affichant une invite de commande et on souhaite explorer et modifier, en lignes de commande, les répertoires et fichiers présents.
On suppose que le répertoire de travail est */home/nsi/terminale*. Pour cette question, on considère que les commandes sont exécutées par l'utilisateur `nsi` et que ce dernier est propriétaire de tous les fichiers et répertoires de l'arborescence */home/nsi*.

(a) Donner la commande permettant d'afficher le répertoire de travail.

(b) Donner l'affichage correspondant à l'utilisation de la commande `ls`.

(c) Donner la commande qui permet de changer le répertoire de travail pour que ce soit le répertoire *histoire*.

Pour la suite de l'exercice, le répertoire de travail est donc */home/nsi/premiere/histoire*. Le résultat de la commande `ls -l` est le suivant :

```
-r--r----- 1 nsi spe  10133 sep. 7 10:07 ada_lovelace.txt
-rw-r----- 1 nsi spe 124158 sep. 5 18:33 turing.jpg
```

(d) Préciser les droits que l'utilisateur `nsi` possède et ceux qu'il ne possède pas sur le fichier **ada_lovelace.txt**.

(e) Écrire la commande permettant de modifier les droits (et ce, récursivement) de l'ensemble des fichiers et répertoires contenus dans le répertoire *histoire*, de telle sorte que l'utilisateur `nsi` ait tous les droits et que tous les autres utilisateurs du système n'en aient aucun.
On supposera dans la suite que l'utilisateur `nsi` a tous les droits dans le répertoire *histoire*.

(f) Écrire la commande qui permet de créer dans le répertoire de travail un répertoire nommé *expose*.

(g) Écrire la commande qui permet de supprimer le fichier **turing.jpg**.

(h) Écrire la commande permettant de déplacer le fichier **ada_lovelace.txt** dans le répertoire *expose*.

7. On suppose dans cette question qu'on dispose d'une salle de TP équipée de machines dont le système d'exploitation est de type Unix.
L'identifiant de l'administrateur et le mot de passe associé sont identiques pour toutes les machines :

identifiant administrateur : adm mot de passe administrateur : Turing

On souhaite, dans cette question, proposer un exercice à destination d'élèves de première NSI leur permettant d'écrire un script Shell dont le but est d'éteindre tous les ordinateurs de la salle informatique à partir d'une machine extérieure à cette salle mais interconnectée aux ordinateurs de la salle informatique grâce à un réseau informatique. On suppose que la connexion se fait en SSH et que celle-ci peut se faire sans mot de passe car la clé publique de l'utilisateur qui va exécuter le script a déjà été diffusée aux machines à éteindre.

(a) Indiquer quelles sont les entrées dont aura besoin le script Shell.

(b) Déterminer les actions que devra comprendre le script Shell envisagé.

(c) Pour chaque action, préciser la ou les commandes Unix correspondantes.

(d) Proposer un énoncé progressif à destination d'élèves de première NSI leur permettant d'aboutir à la construction de ce script.

(e) Proposer une correction à cet énoncé.

8. Votre classe se montre très intéressée par les systèmes d'exploitation. Vous décidez d'aller un peu plus loin sur le sujet par rapport à ce qui est préconisé dans le programme.

(a) Écrire l'énoncé d'un exercice que vous proposeriez à vos élèves afin qu'elles et ils apprennent à utiliser les caractères spéciaux « ? » et « * » dans les commandes Unix.

(b) Indiquer à vos élèves quelles sont les trois entrées-sorties standards.

(c) Donner à vos élèves un exemple de commande qui modifie une de ces entrées-sorties standards.

(d) Donner une commande qui illustre la redirection entre commandes. Expliquer le fonctionnement de cette commande tel que vous le feriez avec vos élèves.

2 Processus

On trouvera dans l'**annexe 5.1** le programme de terminale NSI sur les processus. Dans cette section, on suppose qu'on travaille avec un système Unix conforme au standard POSIX.

2.1 Définitions

9. Donner la définition d'un processus.

10. Un processus est caractérisé par :
- un espace mémoire,
- un ensemble de ressources.

Préciser le sens de ces deux notions.

11. Indiquer la différence fondamentale en terme de gestion de la mémoire entre un processus et un processus léger (thread en anglais).

Dans toute la suite de cette partie, on ne considérera que des processus et aucun processus léger. Sauf mention explicite du contraire, les machines utilisées dans cet exercice ne disposent que d'une seule unité arithmétique et logique (un seul processeur). Ainsi un seul processus peut être exécuté à la fois.

12. Un processus ne peut être lancé que par un autre processus. Donner les noms généralement associés aux deux processus.
13. Donner le nom du seul processus qui ne suit pas cette règle.
14. À un instant t, un processus est caractérisé par son état qui peut être, entre autres, :
    - Prêt ;
    - Bloqué (aussi appelé En attente) ;
    - Élu (aussi appelé En exécution).

    Rappeler en quoi consiste chacun de ces trois états.
15. Proposer une représentation graphique que vous proposeriez à vos élèves pour illustrer les interactions entre ces différents états.

2.2 Visualisation des processus sur un système d'exploitation de type Unix

16. La copie d'écran donnée en figure 1 est le résultat de la commande top lancée dans un terminal.

```
top - 10:39:09 up 9 min,  2 users,  load average: 2,15, 1,88, 1,11
Tâches: 236 total,   2 en cours, 234 en veille,   0 arrêté,   0 zombie
%Cpu(s): 10,4 ut,  3,6 sy,  0,0 ni, 85,9 id,  0,1 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem :   3837,3 total,    736,5 libr,   1301,1 util,   1799,7 tamp/cache
MiB Éch:   2048,0 total,   2048,0 libr,      0,0 util.   1966,6 dispo Mem

    PID UTIL.      PR S  %CPU  %MEM    TEMPS+ COM.                          PPID
   1562 nsi        20 S  16,9   1,6   0:27.69 Xorg                         1560
   1700 nsi        20 R  14,3   6,0   0:35.77 gnome-shell                  1472
   2667 nsi        20 S  11,3   1,8   0:03.99 kazam                        1700
   2710 nsi        20 S   4,0   1,4   0:03.84 thonny                       1700
   2301 nsi        20 S   0,7   1,4   0:02.39 gnome-terminal-              1472
   2974 nsi        20 S   0,7   2,6   0:00.77 WebExtensions                2662
     17 root       20 I   0,3   0,0   0:00.86 kworker/0:1-events             2
     50 root       20 I   0,3   0,0   0:00.96 kworker/2:1-events             2
    129 root       20 I   0,3   0,0   0:00.82 kworker/1:2-events             2
    402 root        0 I   0,3   0,0   0:00.78 kworker/u17:1-i915_flip        2
    429 root       20 I   0,3   0,0   0:00.62 kworker/3:3-events             2
   2160 nsi        20 S   0,3   2,9   0:08.01 okular                       1472
   2178 nsi        20 S   0,3   1,1   0:00.36 kglobalaccel5                1472
   2754 nsi        20 R   0,3   0,1   0:00.19 top                          2745
      1 root       20 S   0,0   0,3   0:02.02 systemd                        0
      2 root       20 S   0,0   0,0   0:00.00 kthreadd                       0
      3 root        0 I   0,0   0,0   0:00.00 rcu_gp                         2
```

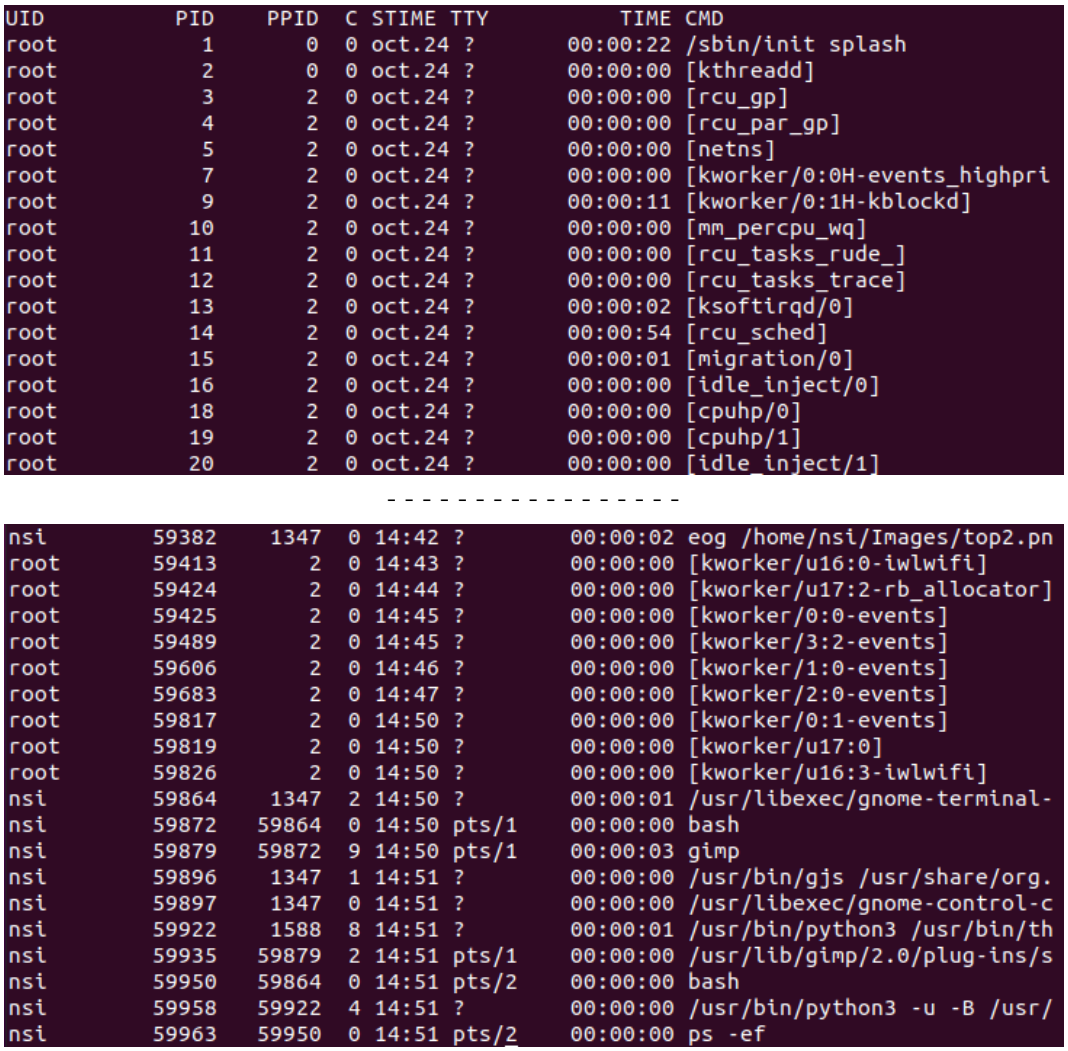
FIGURE 1 – Commande top

(a) Préciser le rôle de la commande top.

(b) Donner la commande à utiliser pour afficher le manuel de la commande top dans un un terminal affichant une invite de commande.

(c) En s'aidant des extraits du manuel de la commande top donnés en **annexe 5.2**, préciser en quoi consiste chacun des descripteurs se trouvant sur la ligne surlignée en blanc commençant par les termes PID UTIL.

17. Proposer une activité à destination des élèves de terminale permettant d'illustrer le principe d'arborescence des processus en utilisant le résultat de la commande `ps -ef` lancée dans un terminal. Un exemple de résultat de cette commande est donné en figure 2.



```
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 oct.24 ?        00:00:22 /sbin/init splash
root           2       0  0 oct.24 ?        00:00:00 [kthreadd]
root           3       2  0 oct.24 ?        00:00:00 [rcu_gp]
root           4       2  0 oct.24 ?        00:00:00 [rcu_par_gp]
root           5       2  0 oct.24 ?        00:00:00 [netns]
root           7       2  0 oct.24 ?        00:00:00 [kworker/0:0H-events_highpri
root           9       2  0 oct.24 ?        00:00:11 [kworker/0:1H-kblockd]
root          10       2  0 oct.24 ?        00:00:00 [mm_percpu_wq]
root          11       2  0 oct.24 ?        00:00:00 [rcu_tasks_rude_]
root          12       2  0 oct.24 ?        00:00:00 [rcu_tasks_trace]
root          13       2  0 oct.24 ?        00:00:02 [ksoftirqd/0]
root          14       2  0 oct.24 ?        00:00:54 [rcu_sched]
root          15       2  0 oct.24 ?        00:00:01 [migration/0]
root          16       2  0 oct.24 ?        00:00:00 [idle_inject/0]
root          18       2  0 oct.24 ?        00:00:00 [cpuhp/0]
root          19       2  0 oct.24 ?        00:00:00 [cpuhp/1]
root          20       2  0 oct.24 ?        00:00:00 [idle_inject/1]
- - - - - - - - - - - - - - - - -
nsi        59382    1347  0 14:42 ?        00:00:02 eog /home/nsi/Images/top2.pn
root       59413       2  0 14:43 ?        00:00:00 [kworker/u16:0-iwlwifi]
root       59424       2  0 14:44 ?        00:00:00 [kworker/u17:2-rb_allocator]
root       59425       2  0 14:45 ?        00:00:00 [kworker/0:0-events]
root       59489       2  0 14:45 ?        00:00:00 [kworker/3:2-events]
root       59606       2  0 14:46 ?        00:00:00 [kworker/1:0-events]
root       59683       2  0 14:47 ?        00:00:00 [kworker/2:0-events]
root       59817       2  0 14:50 ?        00:00:00 [kworker/0:1-events]
root       59819       2  0 14:50 ?        00:00:00 [kworker/u17:0]
root       59826       2  0 14:50 ?        00:00:00 [kworker/u16:3-iwlwifi]
nsi        59864    1347  2 14:50 ?        00:00:01 /usr/libexec/gnome-terminal-
nsi        59872   59864  0 14:50 pts/1    00:00:00 bash
nsi        59879   59872  9 14:50 pts/1    00:00:03 gimp
nsi        59896    1347  1 14:51 ?        00:00:00 /usr/bin/gjs /usr/share/org.
nsi        59897    1347  0 14:51 ?        00:00:00 /usr/libexec/gnome-control-c
nsi        59922    1588  8 14:51 ?        00:00:01 /usr/bin/python3 /usr/bin/th
nsi        59935   59879  2 14:51 pts/1    00:00:00 /usr/lib/gimp/2.0/plug-ins/s
nsi        59950   59864  0 14:51 pts/2    00:00:00 bash
nsi        59958   59922  4 14:51 ?        00:00:00 /usr/bin/python3 -u -B /usr/
nsi        59963   59950  0 14:51 pts/2    00:00:00 ps -ef
```

FIGURE 2 – Résultat de la commande `ps -ef`

2.3 Ordonnancement

L'ordonnancement est l'action qui consiste pour une machine à choisir, au début de chaque unité de temps d'exécution, le processus qui va être exécuté par la machine parmi l'ensemble des différents processus « prêts ». L'ordonnancement est réalisé par l'ordonnanceur du système d'exploitation.

Dans la suite de l'exercice sur le problème d'ordonnancement, on suppose qu'un processus peut être caractérisé par :

- sa **durée d'exécution** exprimée en unités de temps d'exécution (nombre entier) ;
- son **instant d'arrivée** correspondant à l'instant où le processus est déclaré comme « prêt » pour la première fois ;
- sa **position dans la file d'attente** de l'ordonnanceur exprimée par un entier, 1 correspondant à la première position.

Définition 1 (Temps d'attente et temps de réponse)**.**

Deux grandeurs peuvent être calculées sur les processus :

- le temps d'attente qui correspond à la durée écoulée entre l'instant d'arrivée et l'instant de début d'exécution du processus ;
- le temps de réponse qui correspond à la durée écoulée entre l'instant d'arrivée et l'instant de fin d'exécution du processus.

18. Parmi les trois paramètres choisis pour caractériser un processus (durée d'exécution, instant d'arrivée, position dans la file d'attente), indiquer le paramètre qui est *a priori* inconnu de la plupart des systèmes d'exploitation.
19. Proposer un exemple permettant de comprendre l'intérêt d'un ordonnancement intelligent des processus afin d'optimiser leur temps de réponse en s'inspirant de l'ordonnancement de tâches de la vie courante.

2.3.1 Ordonnancement par ordre de soumission

Dans l'ordonnancement par ordre de soumission (premier arrivé, premier servi), les processus « prêts » sont choisis selon l'ordre dans lequel ils arrivent dans la file d'attente de l'ordonnanceur. Le processus choisi s'exécute, soit jusqu'à ce qu'il

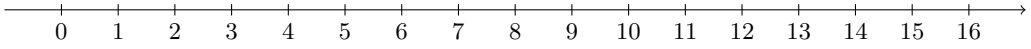
soit terminé, soit jusqu'à ce qu'il se bloque de lui-même car il lui manque une ressource. Il reste bloqué tant qu'il n'a pas obtenu cette ressource.

On donne ci-dessous un exercice à destination d'élèves de terminale NSI.

Exercice 1 (Mise en œuvre de l'ordonnancement par ordre de soumission sur trois processus)**.**
On considère trois processus P1, P2 et P3 soumis à l'ordonnanceur dont les caractéristiques sont les suivantes :

Nom du processus	Durée d'exécution	Instant d'arrivée
P1	8	0
P2	2	5
P3	3	7

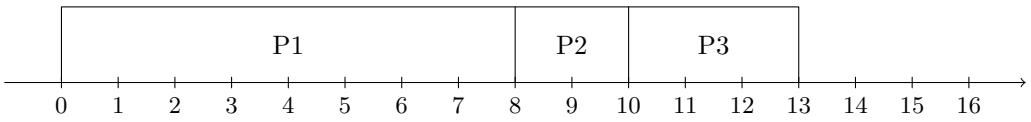
1. Tracer le chronogramme d'exécution de ces trois processus. On supposera que les processus ne se bloquent pas et qu'une unité de temps d'exécution d'un processus correspond à une unité de temps de l'échelle temporelle donnée ci-dessous.



2. Déterminer les temps d'attente et de réponse des trois processus P1, P2 et P3.

20. Voici la réponse attendue par l'enseignant :

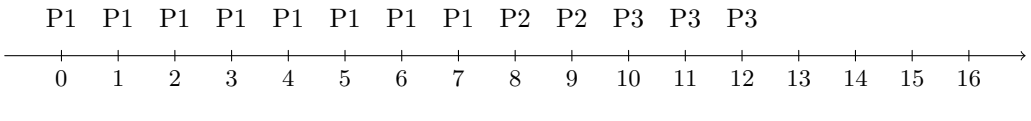
1.



2.
temps d'attente de P1 : 0 temps de réponse de P1 : 8
temps d'attente de P2 : 3 temps de réponse de P2 : 5
temps d'attente de P3 : 3 temps de réponse de P3 : 6

Voici la proposition d'un élève :

1.



2.
temps d'attente de P1 : 0 temps de réponse de P1 : 8
temps d'attente de P2 : 3 temps de réponse de P2 : 4
temps d'attente de P3 : 3 temps de réponse de P3 : 5

Commenter la justesse de sa réponse.

21. Proposer une correction pour cet élève en expliquant pourquoi la représentation choisie par l'enseignant est plus judicieuse que celle proposée par l'élève.

2.3.2 Autres ordonnancements

22. Sur le même principe que l'exercice précédent, proposer une activité à destination d'élèves de terminale NSI permettant d'illustrer les trois différents algorithmes d'ordonnancement décrits ci-dessous.
    On supposera que, dans cet exercice, les configurations des processus seront les mêmes pour les différents ordonnancements et que les processus ne se bloquent pas à cause d'un manque de ressource.
    On donnera également la correction de cette activité.

L'ordonnancement par tourniquet : Le processus élu s'exécute soit pendant un temps donné Q prédéfini appelé quantum (il retourne alors à l'état « prêt » et réintègre la file d'attente de l'ordonnanceur en dernière position), soit jusqu'à ce qu'il soit terminé (durée d'exécution restante inférieure à Q), soit jusqu'à ce qu'il se bloque de lui-même car il lui manque une ressource.

L'ordonnancement par priorité non préemptive : À chaque processus est associé un **numéro de priorité** : le numéro de priorité d'un processus est un entier positif d'autant plus petit que la priorité est grande.

Lorsqu'il doit choisir un processus à élire, l'ordonnanceur choisit celui qui a la priorité la plus grande dans la file d'attente et l'exécute, soit jusqu'à ce qu'il soit terminé, soit jusqu'à ce qu'il se bloque de lui-même car il lui manque une ressource.

L'ordonnancement par priorité préemptive : À chaque processus est associé un **numéro de priorité** : le numéro de priorité d'un processus est un entier positif d'autant plus petit que la priorité est grande.

Au début de chaque unité de temps d'exécution, l'ordonnanceur choisit le processus qui a la priorité la plus grande et l'exécute, ce qui peut provoquer la suspension d'un autre processus en cours de traitement, lequel reprendra lorsqu'il sera le plus prioritaire parmi la file d'attente de l'ordonnanceur.

2.3.3 Prise en compte des ressources

On donne ci-dessous un exercice à destination d'élèves de terminale NSI.

Exercice 2 (Utilisation d'une ressource par plusieurs processus)**.**
On considère trois processus P1, P2 et P3 qui vont être ordonnancés selon la politique du tourniquet avec un quantum Q de 2.
À l'instant initial, ces trois processus ont différentes caractéristiques données ci-dessous :

Nom du processus	Durée d'exécution	Position dans la file d'attente
P1	7	2
P2	6	1
P3	11	3

On a de plus les contraintes suivantes :

- Lors de son exécution, le processus P1 a besoin de la ressource R1 du début de sa 3^e unité de temps d'exécution jusqu'à la fin de sa 7^e unité de temps d'exécution.
- Lors de son exécution, le processus P2 a besoin de la ressource R1 du début de sa 2^e unité de temps d'exécution jusqu'à la fin de sa 5^e unité de temps d'exécution.
- Lors de son exécution, le processus P3 a besoin de la ressource R1 du début de sa 5^e unité de temps d'exécution jusqu'à la fin de sa 9^e unité de temps d'exécution.

1. Tracer le chronogramme d'exécution de ces trois processus
2. Tracer le chronogramme d'allocation de la ressource R1.
3. Donner les temps d'attente et de réponse des trois processus P1, P2 et P3.

23. Proposer une correction de cet exercice permettant de comprendre ce qu'il se passe par une lecture visuelle.
24. Rappeler en quoi consiste le risque d'interblocage.
25. Proposer une activité débranchée permettant de l'illustrer.

2.4 Synthèse

26. Proposer un plan de cours détaillé concernant la gestion des processus et des ressources par un système d'exploitation pour une classe de terminale NSI. Ce plan de cours doit détailler les notions qui seront abordées, la chronologie d'enchaînement de ces notions ainsi qu'une indication sur le temps prévu sur chacune de ces notions. Les éventuels pré-requis ainsi que le type d'activités envisagées seront précisés. Enfin, les objectifs d'apprentissage seront listés.

2.5 Projet : simulation d'un ordonnanceur

Un groupe de trois élèves en classe de terminale NSI souhaite simuler un ordonnanceur en Python permettant d'implémenter les différents ordonnancements vus en cours (et abordés dans la section 2.3 de ce sujet). Ils proposent d'implémenter les processus en programmation orientée objet et souhaitent utiliser une classe `File` pour implémenter la file d'attente de l'ordonnanceur.

27. (a) Une de vos collègues a proposé une implémentation possible d'un processus via la classe suivante :

```
class Processus:
    PRET = 0
    ELU = 1
    BLOQUE = 2
    def __init__(self, nom, priorite, duree_execution, instant_arrivee):
        self.nom = nom
        self.priorite = priorite
        self.duree_execution = duree_execution
        self.instant_arrivee = instant_arrivee
        self.etat = Processus.PRET
```

Commenter cette classe tel que vous le feriez à vos élèves.

(b) Proposer une implémentation de la classe File utilisant la classe standard **list** dont quelques méthodes sont données en **annexe 5.3**. Proposer un exemple d'utilisation de la classe File définie.

28. Les élèves rencontrent des difficultés en voulant programmer l'ordonnancement par priorité préemptive. Après quelques recherches, ils découvrent l'existence d'une classe FileDePriorite sur un forum en ligne.

(a) Proposer une interface pour cette classe.

(b) Proposer une implémentation possible de cette classe utilisant la classe File définie précédemment et la classe standard **dict** dont quelques méthodes sont données en **annexe 5.4**. Proposer un exemple d'utilisation de la classe FileDePriorite définie.

29. Identifier les points d'étapes que vous proposeriez à ce groupe d'élèves pour suivre le projet.

30. On rappelle ci-dessous les compétences constitutives de la pensée informatique proposées dans les programmes de première et terminale NSI :

- analyser et modéliser un problème en termes de flux et de traitement d'informations ;
- décomposer un problème en sous-problèmes ;
- reconnaître des situations déjà analysées et réutiliser des solutions ;
- concevoir des solutions algorithmiques ;
- traduire un algorithme dans un langage de programmation, en spécifier les interfaces et les interactions, comprendre et réutiliser des codes sources existants, développer des processus de mise au point et de validation de programmes ;
- mobiliser les concepts et les technologies utiles pour assurer les fonctions d'acquisition, de mémorisation, de traitement et de diffusion des informations ;
- développer des capacités d'abstraction et de généralisation.

On rappelle ci-après les compétences transversales qui peuvent être développées en NSI :

- faire preuve d'autonomie, d'initiative et de créativité ;
- présenter un problème ou sa solution, développer une argumentation dans le cadre d'un débat ;
- coopérer au sein d'une équipe dans le cadre d'un projet ;
- rechercher de l'information, partager des ressources ;
- faire un usage responsable et critique de l'informatique.

Proposer une grille d'évaluation pour évaluer le travail en projet de ce groupe d'élèves. Vous expliquerez en quoi le projet fait appel à certaines des compétences listées ci-dessus.

Partie B - Gestion de données

Dans ce problème, on se propose d'étudier la gestion des données à travers le continuum des programmes de SNT, première NSI et terminale NSI.

Dans une première partie, on étudie la façon dont sont organisées les données. Dans une seconde partie, on étudie le traitement de ces données.

3 Organisation des données

31. Proposer une activité sans ordinateur à destination d'une classe de SNT permettant d'introduire les concepts de donnée, de descripteur et de collection et conduisant à la notion de données structurées. On précisera le temps prévu pour cette activité.
32. Définir ce que sont les métadonnées d'un fichier et en donner deux exemples.
33. Donner deux exemples de format de fichiers contenant des données structurées.
34. (a) Définir ce qu'est l'indexation de données.
    (b) Proposer une activité sans ordinateur à destination d'une classe de SNT permettant d'expliquer l'indexation de données. On précisera le temps prévu pour cette activité.
35. Pour illustrer l'impact des centres de données (datacenters) sur les pratiques humaines, et pour préparer vos élèves au grand oral, vous décidez d'organiser un débat dans votre cours de SNT en proposant à vos élèves l'exercice suivant :

> **Exercice 3.** Certains magasins proposent de remplacer le ticket de caisse papier par un ticket dématérialisé, envoyé par mail ou sur une application dédiée.
> Êtes-vous pour ou contre cette démarche ? Argumentez votre réponse.

    (a) Donner un argument pour et un argument contre que pourraient proposer vos élèves pour défendre chacun des deux points de vue.
    (b) On rappelle ci-après les cinq compétences évaluées lors du Grand Oral :
        - qualité de la prise de parole en continu ;
        - qualité orale ;
        - qualité des connaissances ;
        - qualité et construction de l'argumentation ;
        - qualité de l'interaction.

        Dans l'optique de préparer vos élèves à cette épreuve, vous en interrogez deux pour débattre à l'oral sur cet exercice, chacun défendant un point de vue qui lui est imposé. Proposer une grille d'évaluation avec un barème précis s'appuyant sur ces cinq compétences.
36. En classe de première NSI, le programme demande de savoir importer une table depuis un fichier texte tabulé ou un fichier CSV (voir **annexe 5.5**).
    (a) En Python, recommanderiez-vous l'usage d'une bibliothèque de ce langage à cette fin ? Pourquoi, et si oui, laquelle ?
    (b) Dans le programme de première NSI, dans la partie traitement de données en tables, il est précisé : « Est utilisé un tableau doublement indexé ou un tableau de p-uplets qui partagent les mêmes descripteurs » (voir **annexe 5.5**). On précise que les p-uplets en question sont des p-uplets nommés et que, d'après le programme de première NSI, « en Python, les p-uplets nommés sont implémentés par des dictionnaires ».
        Donner un avantage et un inconvénient d'utiliser un tableau doublement indexé et un avantage et un inconvénient d'utiliser un tableau de p-uplets nommés qui partagent les mêmes descripteurs pour importer une table en Python.
37. (a) En terminale NSI, un élève a proposé une organisation des données dans une seule table (voir figure 3). Cette table lui permet de recueillir, entre autres, la date, l'heure et la salle de diffusion de films d'un complexe cinématographique.
        Donner trois arguments pour lui faire comprendre les inconvénients d'une telle organisation.

Titre	Réalisateur	Année	Durée	Salle	Heure	Jour
De battre mon cœur s'est arrêté	Jacques Audiard	2005	107	4	14 :40	2021-06-12
Le Cercle des poètes disparus	Peter Weir	1989	128	5	20 :30	2021-06-12
...	...	...	...	...	...	...

FIGURE 3 – Proposition d'un élève

    (b) Proposer une autre structure pour cette base de données permettant de pallier ces inconvénients.
38. (a) Le modèle relationnel est au programme de terminale NSI (voir **annexe 5.6**).
        Proposer un plan de cours succint sur ce sujet, tel que vous le présenteriez à des élèves de terminale NSI.
    (b) Proposer une activité permettant d'introduire la notion de contrainte d'intégrité.
39. Définir, tel que vous le feriez à une classe de terminale NSI, ce qu'est un SGBD et son rôle.

4 Traitement des données

40. En SNT, le programme donne l'exemple d'activité suivante : « Télécharger des données ouvertes (sous forme d'un fichier au format CSV avec les métadonnées associées), observer les différences de traitements possibles selon le logiciel choisi pour lire un fichier : programme Python, tableur, éditeur de textes ou encore outils spécialisés en ligne. »
    (a) Proposer deux sites sur lesquels on peut télécharger des données ouvertes.
    (b) Citer un avantage et un inconvénient pour chacune des quatre propositions (programme Python, tableur, éditeur de textes ou outils spécialisés en ligne) permettant de lire le fichier CSV.
    (c) Choisir deux logiciels puis proposer une activité permettant d'observer les différences de traitements possibles selon le logiciel choisi pour lire le fichier.

41. En classe de première NSI, vous décidez de faire travailler les élèves sur trois fichiers CSV, contenant respectivement des données sur les régions françaises, les départements français et les communes françaises.
    Vous avez utilisé trois tableaux de dictionnaires qui partagent les mêmes clés pour importer trois tables depuis ces trois fichiers CSV.
    Pour se donner une idée :
    - un élément du tableau `REGION` est
      `{"CodeRegion": 84, "NomRegion": "Auvergne-Rhône-Alpes","CodePref": 69123, "CodeCR": "69123"}`
    - un élément du tableau `DEPARTEMENT` est
      `{"NumDep": 1, "CodeRegion": 84, "CodePref": 1053, "NomDep": "Ain"}`
    - un élément du tableau `COMMUNE` est
      `{"NumDep": 1, "NomVille": "Nantua", "CodePostal": "01460", "CodeInsee": "01269", "Population": 3713}`

    Les valeurs des clés `"CodePref"` et `"CodeCR"` des éléments des tableaux `REGION` et `DEPARTEMENT` correspondent aux valeurs de la clé `"CodeInsee"` des éléments du tableau `COMMUNE`.

    À partir des trois tableaux `REGION`, `DEPARTEMENT` et `COMMUNE`, vous proposez à vos élèves de première NSI l'exercice suivant :

    Exercice
    1) Écrire une fonction Python renvoyant un tableau contenant le nom des communes de plus de 10 000 habitants.

    2) Écrire une fonction Python renvoyant le nombre de départements ayant une commune dont le nom est `"Saint-Michel"`.

    3) Écrire une fonction Python renvoyant un tableau de p-uplets contenant le nom des régions et le nom de leurs préfectures de régions lorsque la préfecture de région a plus de 250 000 habitants.

    (a) Identifier pour chaque question de l'exercice une difficulté algorithmique à laquelle on peut s'attendre de la part d'un élève de première.
    (b) Pour chacune des difficultés identifiées dans la question précédente, proposer une activité de remédiation permettant de la surmonter.
    (c) Proposer deux corrections de cet exercice :
        - l'une utilisant des boucles ;
        - l'autre utilisant uniquement des tableaux définis par compréhension.

42. On dispose d'une base de données pour gérer les locations de VTT électriques d'un magasin de sport. Le schéma relationnel de cette base de données est :
    VTT(<u>num_vtt</u>, `marque`, `prix_achat`, `date_achat`)
    CLIENT(<u>num_client</u>, `nom`, `prenom`, `adresse_mail`)
    LOCATION(<u>id_loc</u>, `num_VTT#`, `num_client#`, `date_location`, `date_retour`)

    `date_retour` peut être égal à `NULL` si le VTT est en cours de location.

    Vous utilisez cette base de données pour construire un exercice permettant d'évaluer l'ensemble des notions sur le langage SQL au programme de terminale NSI (voir **annexe 5.6**).
    (a) Dans la convention choisie par le concepteur du schéma relationnel, préciser à quoi correspond les descripteurs soulignés et les descripteurs suivis d'un `#`.
    (b) Rédiger l'énoncé de cet exercice à partir de cette base de données. Les questions devront être progressives, non redondantes et respecter le programme de terminale NSI.
        Pour chaque question, déterminer ce qu'elle évalue.
    (c) Proposer une correction de cet exercice.
    (d) Proposer une grille d'évaluation de cet exercice qui indique avec précision les éléments évalués.

5 Annexes

5.1 Programme de terminale NSI sur les processus

Architectures matérielles, systèmes d'exploitation et réseaux

La réduction de taille des éléments des circuits électroniques a conduit à l'avènement de systèmes sur puce (*SoCs* pour *Systems on Chips* en anglais) qui regroupent dans un seul circuit nombre de fonctions autrefois effectuées par des circuits séparés assemblés sur une carte électronique. Un tel système sur puce est conçu et mis au point de façon logicielle, ses briques électroniques sont accessibles par des API, comme pour les bibliothèques logicielles.

Toute machine est dotée d'un système d'exploitation qui a pour fonction de charger les programmes depuis la mémoire de masse et de lancer leur exécution en leur créant des processus, de gérer l'ensemble des ressources, de traiter les interruptions ainsi que les entrées-sorties et enfin d'assurer la sécurité globale du système.

Dans un réseau, les routeurs jouent un rôle essentiel dans la transmission des paquets sur Internet : les paquets sont routés individuellement par des algorithmes. Les pertes logiques peuvent être compensées par des protocoles reposant sur des accusés de réception ou des demandes de renvoi, comme TCP.

La protection des données sensibles échangées est au cœur d'Internet. Les notions de chiffrement et de déchiffrement de paquets pour les communications sécurisées sont explicitées.

Contenus	**Capacités attendues**	**Commentaires**
Composants intégrés d'un système sur puce.	Identifier les principaux composants sur un schéma de circuit et les avantages de leur intégration en termes de vitesse et de consommation.	Le circuit d'un téléphone peut être pris comme un exemple : microprocesseurs, mémoires locales, interfaces radio et filaires, gestion d'énergie, contrôleurs vidéo, accélérateur graphique, réseaux sur puce, etc.
Gestion des processus et des ressources par un système d'exploitation.	Décrire la création d'un processus, l'ordonnancement de plusieurs processus par le système. Mettre en évidence le risque de l'interblocage *(deadlock)*.	À l'aide d'outils standard, il s'agit d'observer les processus actifs ou en attente sur une machine. Une présentation débranchée de l'interblocage peut être proposée.

5.2 Extraits du manuel de la commande top

```
1. %CPU  --  CPU Usage
   The  task's share of the elapsed CPU time since the last screen up-
   date, expressed as a percentage of total CPU time.

   In a true SMP environment, if a process is multi-threaded  and  top
   is  not operating in Threads mode, amounts greater than 100% may be
   reported.  You toggle Threads mode with the  `H'  interactive  com-
   mand.

   Also  for  multi-processor  environments,  if Irix mode is Off, top
   will operate in Solaris mode where a task's cpu usage will  be  di-
   vided  by  the total number of CPUs.  You toggle Irix/Solaris modes
   with the `I' interactive command.

   Note: When running in forest view mode  (`V')  with  children  col-
   lapsed  (`v'),  this  field will also include the CPU time of those
   unseen children.  See topic 4c. TASK  AREA  Commands,  CONTENT  for
   more information regarding the `V' and `v' toggles.

2. %MEM  --  Memory Usage (RES)
   A task's currently resident share of available physical memory.

   See `OVERVIEW, Linux Memory Types' for additional details.
```

```
19. PID  --  Process Id
    The  task's  unique  process  ID,  which periodically wraps, though
    never restarting at zero.  In kernel terms, it  is  a  dispatchable
    entity defined by a task_struct.

    This  value  may  also be used as: a process group ID (see PGRP); a
    session ID for the session leader (see SID); a thread group ID  for
    the  thread group leader (see TGID); and a TTY process group ID for
    the process group leader (see TPGID).

20. PPID  --  Parent Process Id
    The process ID (pid) of a task's parent.

21. PR  --  Priority
    The scheduling priority of the task.   If  you  see  `rt'  in  this
    field, it means the task is running under real time scheduling pri-
    ority.

    Under linux, real time priority is somewhat misleading since tradi-
    tionally  the  operating itself was not preemptible.  And while the
    2.6 kernel can be made mostly preemptible, it is not always so.
```

```
29. S  --  Process Status
    The status of the task which can be one of:
        D = uninterruptible sleep
        I = idle
        R = running
        S = sleeping
        T = stopped by job control signal
        t = stopped by debugger during trace
        Z = zombie

    Tasks  shown as running should be more properly thought of as ready
    to run  --  their task_struct is simply represented  on  the  Linux
    run-queue.   Even  without a true SMP machine, you may see numerous
    tasks in this state depending on  top's  delay  interval  and  nice
    value.
```

38. **TIME** -- CPU Time
Total CPU time the task has used since it started. When Cumulative mode is On, each process is listed with the cpu time that it and its dead children have used. You toggle Cumulative mode with `S', which is both a command-line option and an interactive command. See the `S' interactive command for additional information regarding this mode.

39. **TIME+** -- CPU Time, hundredths
The same as TIME, but reflecting more granularity through hundredths of a second.

6. **COMMAND** -- Command **Name** or Command **Line**
Display the command line used to start a task or the name of the associated program. You toggle between command line and name with `c', which is both a command-line option and an interactive command.

When you've chosen to display command lines, processes without a command line (like kernel threads) will be shown with only the program name in brackets, as in this example:
[kthreadd]

This field may also be impacted by the forest view display mode. See the `V' interactive command for additional information regarding that mode.

Note: The COMMAND field, unlike most columns, is not fixed-width. When displayed, it plus any other variable width columns will be allocated all remaining screen width (up to the maximum 512 characters). Even so, such variable width fields could still suffer truncation. This is especially true for this field when command lines are being displayed (the `c' interactive command.) See topic 5c. SCROLLING a Window for additional information on accessing any truncated data.

44. **USER** -- User Name
The effective user name of the task's owner.

5.3 Extraits de l'aide Python pour les objets de type list

```
>>> help(len)
 Help on built-in function len in module builtins:

 len(obj, /)
     Return the number of items in a container.

>>> help(list)
 Help on class list in module builtins:

 class list(object)
  |  list(iterable=(), /)
  |
  |  Built-in mutable sequence.
  |
  |  If no argument is given, the constructor creates a new empty list.
  |  The argument must be an iterable if specified.
  |
  |  Methods defined here:
  |
  |  __len__(self, /)
  |      Return len(self).
  |
  |  append(self, object, /)
  |      Append object to the end of the list.
  |
  |  clear(self, /)
  |      Remove all items from list.
  |
  |  copy(self, /)
  |      Return a shallow copy of the list.
  |
  |  index(self, value, start=0, stop=9223372036854775807, /)
  |      Return first index of value.
  |
  |      Raises ValueError if the value is not present.
  |
  |  insert(self, index, object, /)
  |      Insert object before index.
  |
  |  pop(self, index=-1, /)
  |      Remove and return item at index (default last).
  |
  |      Raises IndexError if list is empty or index is out of range.
  |
  |  remove(self, value, /)
  |      Remove first occurrence of value.
  |
  |      Raises ValueError if the value is not present.
  |
  |  reverse(self, /)
  |      Reverse *IN PLACE*.
  |
  |  sort(self, /, *, key=None, reverse=False)
  |      Sort the list in ascending order and return None.
  |
  |      The sort is in-place (i.e. the list itself is modified) and stable (i.e.
  |      the order of two equal elements is maintained).
  |
  |  ----------------------------------------------------------------------
  |  Data and other attributes defined here:
  |
  |  __hash__ = None
```

5.4 Extraits de l'aide Python pour les objets de type dict

```
>>> help(min)
 Help on built-in function min in module builtins:

 min(...)
     min(iterable, *[, default=obj, key=func]) -> value
     min(arg1, arg2, *args, *[, key=func]) -> value

     With a single iterable argument, return its smallest item. The
     default keyword-only argument specifies an object to return if
     the provided iterable is empty.
     With two or more arguments, return the smallest argument.

>>> help(dict)
 Help on class dict in module builtins:

 class dict(object)
  |  dict() -> new empty dictionary
  |  dict(mapping) -> new dictionary initialized from a mapping object's
  |      (key, value) pairs
  |  dict(iterable) -> new dictionary initialized as if via:
  |      d = {}
  |      for k, v in iterable:
  |          d[k] = v
  |
  |  Methods defined here:
  |
  |  __contains__(self, key, /)
  |      True if the dictionary has the specified key, else False.
  |
  |  __delitem__(self, key, /)
  |      Delete self[key].
  |
  |  __init__(self, /, *args, **kwargs)
  |      Initialize self.  See help(type(self)) for accurate signature.
  |
  |  __iter__(self, /)
  |      Implement iter(self).
  |
  |  __len__(self, /)
  |      Return len(self).
  |
  |  clear(...)
  |      D.clear() -> None.  Remove all items from D.
  |
  |  copy(...)
  |      D.copy() -> a shallow copy of D
  |
  |  items(...)
  |      D.items() -> a set-like object providing a view on D's items
  |
  |  keys(...)
  |      D.keys() -> a set-like object providing a view on D's keys
  |
  |  pop(...)
  |      D.pop(k[,d]) -> v, remove specified key and return the corresponding value.
  |      If key is not found, d is returned if given, otherwise KeyError is raised
  |
  |  ----------------------------------------------------------------------
  |  Data and other attributes defined here:
  |
  |  __hash__ = None
```

5.5 Programme de première NSI sur les données en table

Traitement de données en tables

Les données organisées en table correspondent à une liste de p-uplets nommés qui partagent les mêmes descripteurs. La mobilisation de ce type de structure de données permet de préparer les élèves à aborder la notion de base de données qui ne sera présentée qu'en classe terminale. Il s'agit d'utiliser un tableau doublement indexé ou un tableau de p-uplets, dans un langage de programmation ordinaire et non dans un système de gestion de bases de données.

Contenus	Capacités attendues	Commentaires
Indexation de tables	Importer une table depuis un fichier texte tabulé ou un fichier CSV.	Est utilisé un tableau doublement indexé ou un tableau de p-uplets qui partagent les mêmes descripteurs.
Recherche dans une table	Rechercher les lignes d'une table vérifiant des critères exprimés en logique propositionnelle.	La recherche de doublons, les tests de cohérence d'une table sont présentés.
Tri d'une table	Trier une table suivant une colonne.	Une fonction de tri intégrée au système ou à une bibliothèque peut être utilisée.
Fusion de tables	Construire une nouvelle table en combinant les données de deux tables.	La notion de domaine de valeurs est mise en évidence.

5.6 Programme de terminale NSI sur les bases de données

Bases de données

Le développement des traitements informatiques nécessite la manipulation de données de plus en plus nombreuses. Leur organisation et leur stockage constituent un enjeu essentiel de performance.

Le recours aux bases de données relationnelles est aujourd'hui une solution très répandue. Ces bases de données permettent d'organiser, de stocker, de mettre à jour et d'interroger des données structurées volumineuses utilisées simultanément par différents programmes ou différents utilisateurs. Cela est impossible avec les représentations tabulaires étudiées en classe de première.

Des systèmes de gestion de bases de données (SGBD) de très grande taille (de l'ordre du pétaoctet) sont au centre de nombreux dispositifs de collecte, de stockage et de production d'informations.

L'accès aux données d'une base de données relationnelle s'effectue grâce à des requêtes d'interrogation et de mise à jour qui peuvent par exemple être rédigées dans le langage SQL (*Structured Query Language*). Les traitements peuvent conjuguer le recours au langage SQL et à un langage de programmation.

Il convient de sensibiliser les élèves à un usage critique et responsable des données.

Contenus	**Capacités attendues**	**Commentaires**
Modèle relationnel : relation, attribut, domaine, clef primaire, clef étrangère, schéma relationnel.	Identifier les concepts définissant le modèle relationnel.	Ces concepts permettent d'exprimer les contraintes d'intégrité (domaine, relation et référence).
Base de données relationnelle.	Savoir distinguer la structure d'une base de données de son contenu. Repérer des anomalies dans le schéma d'une base de données.	La structure est un ensemble de schémas relationnels qui respecte les contraintes du modèle relationnel. Les anomalies peuvent être des redondances de données ou des anomalies d'insertion, de suppression, de mise à jour. On privilégie la manipulation de données nombreuses et réalistes.
Système de gestion de bases de données relationnelles.	Identifier les services rendus par un système de gestion de bases de données relationnelles : persistance des données, gestion des accès concurrents, efficacité de traitement des requêtes, sécurisation des accès.	Il s'agit de comprendre le rôle et les enjeux des différents services sans en détailler le fonctionnement.
Langage SQL : requêtes d'interrogation et de mise à jour d'une base de données.	Identifier les composants d'une requête. Construire des requêtes d'interrogation à l'aide des clauses du langage SQL : SELECT, FROM, WHERE, JOIN. Construire des requêtes d'insertion et de mise à jour à l'aide de : UPDATE, INSERT, DELETE.	On peut utiliser DISTINCT, ORDER BY ou les fonctions d'agrégation sans utiliser les clauses GROUP BY et HAVING.