

	<p align="center"><b>Examen Session Normale</b> <b>12/01/2023</b></p>	
A. Bertails, N.A. Diago, T. Guidini, J.A. Lorenzo	Système d'exploitation	
ING1 Informatique-Mathématiques appliquées	Année 2022–2023	
ING1 Informatique-Mathématiques Apprentissage		

### Modalités

- Durée : 2 heures.
- Vous devez rédiger votre copie à l'aide d'un stylo à encre exclusivement.
- Toutes vos affaires (sacs, vestes, trousse, etc.) doivent être placées à l'avant de la salle.
- Aucun document n'est autorisé.
- La calculatrice est autorisée.
- Le QCM peut contenir une ou plusieurs réponses valides.
- Aucune question ne peut être posée aux enseignants, posez des hypothèses en cas de doute.
- Aucune sortie n'est autorisée avant une durée incompressible d'une heure.
- Aucun déplacement n'est autorisé.

### QCM à répondre sur votre copie et non pas sur le sujet : (6 points)

- a) La *swap* ou espace d'échange...
- ☐ est disponible à l'utilisateur sur le point de montage /swap
  - ☐ évite l'utilisation du système de pagination
  - ☐ contient l'installation du système d'exploitation
  - ☐ **Aucune des réponses n'est valide**
- b) Pendant le processus de traduction d'une adresse virtuelle, si le bit de validité (V) dans la table de pages est à 0, nous avons que :
- ☐ la page est en mémoire réelle
  - ☐ **un défaut de page est déclenché**
  - ☐ le système choisira un algorithme de remplacement de pages
  - ☐ Aucune des réponses n'est valide

- c) Nous souhaitons installer un paquet \*.deb que nous avons téléchargé de l'internet, car il n'est pas disponible sur les dépôts officiels. Quelle commande devons-nous utiliser ?
- ☐ add-apt-repository
  - ☐ apt install
  - ☒ **dpkg**
  - ☐ Aucune des réponses n'est valide
- d) Un conteneur Docker...
- ☒ **n'a pas de kernel, il utilise le kernel du host**
  - ☐ nécessite un hyperviseur
  - ☐ n'a pas besoin d'une image
  - ☐ Aucune des réponses n'est valide
- e) Virtualbox est un hyperviseur
- ☐ de Type I
  - ☒ **de Type II**
  - ☐ de Type III
  - ☐ Aucune des réponses n'est valide
- f) Comment pouvons-nous classer les interruptions attendues par le processeur ?
- ☐ Externes (déroutements) et internes (matérielles et logicielles)
  - ☒ **Externes (matérielles et logicielles) et internes (déroutements)**
  - ☐ Internes (logicielles) et externes (matérielles)
  - ☐ Internes (matérielles) et externes (logicielles)

### Questions de cours : (5 points)

a) Expliquer les fonctionnalités fournies par le Gestionnaire de Système de Fichiers Virtuels ainsi que son rôle dans le fonctionnement du système d'exploitation.

Un **Système de Gestion de Fichiers** (SGF) est une structure de données permettant de stocker les informations et de les organiser dans des fichiers sur des mémoires secondaires. Le SGF est la partie la plus visible d'un SE. Le SGF se charge de gérer le stockage et la manipulation des fichiers sur le matériel physique. Il offre également les primitives pour manipuler les fichiers. Les tâches effectuées par le SGF :

- Fournit une interface "conviviale" pour faciliter la manipulation des fichiers
- Gestion de l'organisation des fichiers sur le disque (allocation de l'espace)

- disque aux fichiers)
- Gestion de l'espace libre
- Gestion des fichiers dans un environnement multi-utilisateurs

b) Expliquer la différence entre mémoire cache et mémoire centrale.

L'objectif de la **Mémoire Cache** est d'éviter de rechercher en mémoire centrale des données déjà cherchées précédemment en les conservant près du processeur dans une petite mémoire à accès rapide. Géré par le matériel (hors SE). Principe de localité :

- Localité temporelle : tendance à réutiliser des données récemment accédées (instructions d'une boucle);
- Localité spatiale : tendance à référencer des données voisines d'autres données récemment accédées (tableau  $a[i]$ ,  $a[i+1]$ ,...).

**La mémoire centrale ou vive (ou RAM)** est la mémoire dans laquelle les données et processus sont placées lors de leur traitement. Quelques caractéristiques :

- mémoire à accès rapide ;
- mémoire volatile i.e., besoin de courant électrique ;
- "cases" numérotées à partir de 0 (le numéro correspond à l'adresse ; chaque case contient un octet (8 bits; 1 bit : 1 ou 0) - 1 octet = 8 bits →  $2^8 = 256$  valeurs possibles) ;
- deux types d'opérations possibles : lecture / écriture

c) Expliquer la différence entre machine virtuelle et conteneur docker.

Une **machine virtuelle ou VM** est un environnement entièrement virtualisé qui fonctionne sur une machine physique. Elle exécute son propre système d'exploitation (OS) et bénéficie des mêmes équipements qu'une machine physique : CPU, mémoire RAM, disque dur et carte réseau.

**Docker est un logiciel libre permettant de gérer des conteneurs.** Un conteneur est une abstraction qui regroupe le code et toutes ses dépendances afin que l'application s'exécute rapidement et de manière fiable d'un environnement informatique à un autre. Un conteneur est l'exécution d'une image : il possède la copie du système de fichiers de l'image, ainsi que la capacité de lancer des processus.

d) Quel est l'avantage des paquets *Snap* et *Flatpak* par rapport aux paquets classiques *.deb* ou *.rpm* ?

**Snap** est un système d'emballage et de déploiement de logiciels pour les systèmes d'exploitation utilisant le noyau Linux. Les paquets, appelés *snaps*, et l'outil pour les utiliser, *snapd*, fonctionnent sur une gamme de distributions Linux et permettent aux développeurs de logiciels de distribuer leurs applications directement aux utilisateurs. Les snaps sont des applications autonomes qui s'exécutent dans un bac à sable avec un accès modéré au système hôte.

**Flatpak** (anciennement *xdg-app*) est un système de virtualisation d'application pour les distributions GNU/Linux de bureau.

Le **dpkg** est utilisé pour installer, supprimer et fournir des informations à propos des paquets *\*.deb* qui sont supportés par les distributions basées Debian. C'est un outil de bas niveau.

Le **rpm** permet d'installer, mettre à jour, désinstaller, vérifier et rechercher des paquets, avec les droits de l'utilisateur root.

e) Quelles sont les différences et le fonctionnement des commandes *apt-get upgrade*, *apt upgrade*, *apt-get dist-upgrade* et *do-release-upgrade* ?

### **apt-get upgrade**

La commande upgrade permet d'installer les versions les plus récentes de tous les paquets présents sur le système en utilisant les sources énumérées dans /etc/apt/sources.list. Les paquets installés dont il existe de nouvelles versions sont récupérés et mis à niveau. En aucun cas des paquets déjà installés ne sont supprimés ; de même, des paquets qui ne sont pas déjà installés ne sont ni récupérés ni installés.

### **apt upgrade**

Mets à jour les paquets installés sans en supprimer. Installe de nouveaux paquets pour satisfaire des dépendances si nécessaire.

### **apt-get dist-upgrade**

La commande dist-upgrade effectue la fonction upgrade en y ajoutant une gestion intelligente des changements de dépendances dans les nouvelles versions des paquets ; apt-get possède un système « intelligent » de résolution des conflits et il essaye, quand c'est nécessaire, de mettre à niveau les paquets les plus importants aux dépens des paquets les moins importants. Le fichier /etc/apt/sources.list contient une liste de sources où récupérer les paquets désirés.

### **do-release-upgrade**

Démarre le upgrade d'une nouvelle version du SE, i.e., met à niveau Ubuntu vers une nouvelle version.

## **Exercice 1 : Programmation de processus (1.5 points)**

Concernant le programme ci-dessous, répondre aux questions suivantes :

- a) Quelles sont les lignes affichées par le programme ? **(0.75 points)**
- b) Quels sont les ordres possibles de ces lignes ? **(0.75 points)**

**Note :** l'opérateur "!" dans la ligne 31 correspond à la négation. C'est à dire :

Pour  $pid = 0 \rightarrow !pid = 1$   
Pour  $pid \neq 0 \rightarrow !pid = 0$

```

1 #include <unistd.h>      /* Symbolic Constants */
2 #include <sys/types.h>   /* Primitive System Data Types */
3 #include <errno.h>       /* Errors */
4 #include <stdio.h>       /* Input/Output */
5 #include <sys/wait.h>    /* Wait for Process Termination */
6 #include <stdlib.h>      /* General Utilities */
7 #include <string.h>      /* String utilities*/
8
9 pid_t Fork(void)
10 {
11     pid_t pid;
12
13     pid = fork();
14
15     if (pid < 0){
16         printf("Fork error");
17         exit(-1);
18     }
19
20     return pid;
21 }
22
23
24 int main()
25 {
26     int status;
27     pid_t pid;
28
29     printf("Hello\n");
30     pid = Fork();
31     printf("%d\n", pid);
32     if (pid != 0) {
33         if (wait(&status) > 0) {
34             if (WIFEXITED(status) != 0)
35                 printf("%d\n", WEXITSTATUS(status));
36         }
37     }
38     printf("Bye\n");
39     exit(2);
40 }
41

```

### Réponse :

Deux sorties possibles :

Si le père affiche sa sortie avant le fils :

Hello

0

1

Bye

2

Bye

Si le fils affiche sa sortie avant le père :

Hello

1

0

Bye

2

Bye

## Exercice 2 : Le système de fichiers (1.5 points)

On considère un système de fichiers tel que l'information concernant les blocs de données de chaque fichier est donc accessible à partir du i-noeud de celui-ci (comme dans UNIX). On supposera que :

- Le système de fichiers utilise des blocs de données de taille fixe 1K (1024 octets).
- L'i-noeud de chaque fichier (ou répertoire) contient 12 pointeurs directs sur des blocs de données, 1 pointeur indirect simple, 1 pointeur indirect double et 1 pointeur indirect triple.
- Chaque pointeur (numéro de bloc) est représenté sur 4 octets.

a) Quelle est la plus grande taille de fichier que ce système de fichiers peut supporter ? **(0.75 points)**

**$12 \times 1\text{ko} + 256 \times 1\text{ko} + 256 \times 256 \times 1\text{ko} + 256 \times 256 \times 256 \times 1\text{ko}$**

b) On considère un fichier contenant 100,000 octets. Combien de blocs de données sont-ils nécessaires (au total) pour représenter ce fichier sur disque ? **(0.75 points)**

**taille du fichier divisé par le taille d'un bloc =  $100000 / 1024 = 97$  ou  $98$  blocs**

## Exercice 3 : Ordonnancement (3 points)

Considérons cinq processus P1, P2, P3, P4 et P5, dont les temps d'exécution et leurs temps d'arrivée respectifs sont donnés dans le tableau ci-dessous.

Processus	Temps d'exécution	Temps d'arrivée
P1	3	0
P2	6	2
P3	4	4
P4	5	6
P5	2	8

Pour chaque technique d'ordonnancement, FCFS (First Come First Served), RR (Round Robin, quantum = 4 unités de temps), SJF (Shortest Job First, sans préemption) et SRT (Shortest Remaining Time) :

- Faire un schéma qui illustre l'exécution de ces différents processus. **(0.25 point pour chaque)**
- Calculer le temps de séjour de chaque processus (intervalle de temps entre la soumission du processus et son achèvement), le temps moyen de séjour, le temps d'attente (temps de séjour - temps d'exécution du processus) et le temps moyen d'attente. **(0.50 point pour chaque)**

FCFS

0					5					10					15					20
P1	X	X	X																	
P2				X	X	X	X	X	X											
P3										X	X	X	X							
P4														X	X	X	X	X		
P5																			X	X

Processus	Temps de séjour	Temps d'attente
P1	3	0
P2	7	1
P3	9	5
P4	12	7
P5	12	10
Moyenne	8.6	4.6

RR (quantum = 4 unités de temps)

0					5					10					15					20
P1	X	X	X																	
P2				X	X	X	X										X	X		
P3								X	X	X	X									
P4											X	X	X	X						X
P5															X	X				

Processus	Temps de séjour	Temps d'attente
P1	3	0
P2	17	11
P3	7	3
P4	14	9
P5	9	7
Moyenne	10	6

SJF

0					5					10					15					20
P1	X	X	X																	
P2				X	X	X	X	X	X											
P3												X	X	X	X					
P4																X	X	X	X	X
P5										X	X									

Processus	Temps de séjour	Temps d'attente
P1	3	0
P2	7	1
P3	11	7
P4	14	9
P5	3	1
Moyenne	7.6	3.6



## SRT

0					5					10					15					20
P1	X	X	X																	
P2				X							X	X	X	X	X					
P3					X	X	X	X												
P4																X	X	X	X	X
P5									X	X										

Processus	Temps de séjour	Temps d'attente
P1	3	0
P2	13	7
P3	4	0
P4	14	9
P5	2	0
Moyenne	7.2	3.2

### Exercice 4 : Mémoire virtuelle (3 points)

Un ordinateur possède un système de mémoire virtuelle avec pagination qui utilise des pages de 8 KB. Son espace de mémoire virtuelle est de  $2^{32}$  bytes et la mémoire physique a une taille de  $2^{23}$  bytes. La table de pages correspondante à un processus en exécution est la suivante :

Num. page	Cadre de page
0	1
1	7
2	9
3	14
4	8
5	3
6	25
7	16
8	23
9	78

Répondre aux questions suivantes :

- Quel est le format de l'adresse virtuelle ? Indiquer chaque champ et le nombre de bits de chacun. **(1 point)**

**Pages de 8KB =  $2^{13}$  bytes. 32 - 13 = 19 bits supérieures pour le numéro de page et 13 inférieures pour l'offset ou déplacement.**

b) Indiquer l'adresse physique correspondante à l'adresse virtuelle : **(1 point)**

0000 0000 0000 0000 0110 0000 1000 1010

**Dans l'adresse, les 19 bits du numéro de page sont**

**0000 0000 0000 0000 011 = 3**

**donc page 3, en mémoire, qui se trouve dans le cadre de page 14 selon la table de pages. 14 en binaire = 1110. Donc l'adresse physique est :**

**0000 0000 0000 0001 1100 0000 1000 1010**

c) Quelle est la taille de mémoire occupée par l'espace d'adresses virtuelles de ce processus ? **(0.5 point)**

**Le processus a une table de 10 pages, donc 10 pages \* 8 KB/page = 80 KB.**

d) Quelle est la taille en MB de la mémoire physique ? **(0.5 point)**

**La mémoire physique a une taille de  $2^{23}$  bytes =  $2^{13}$  KB =  $2^3$  MB = 8 MB.**