

# Module SY5 – Systèmes d'Exploitation

Dominique Poulalhon

`dominique.poulalhon@irif.fr`

Université de Paris (Diderot)

L3 Informatique & DL Bio-Info, Jap-Info, Math-Info

Année universitaire 2021-2022

## ÉQUIPE ENSEIGNANTE

Responsable du cours : Dominique Poulalhon

dominique.poulalhon@irif.fr

Chargés de TP

- Groupe INFO 1 (jeudi 10h45 – salle **2001**) : Colin Gonzalez  
gonzalez@irif.fr
- Groupe INFO 2 (mardi 16h15) : Guillaume Geoffroy  
guillaume.geoffroy@irif.fr
- Groupe INFO 3 (lundi 14h) : Dominique Poulalhon
- Groupe INFO 4 (jeudi 10h45) : Guillaume Geoffroy
- Groupe INFO 5 (lundi 16h15) : Anne Micheli et Dominique Poulalhon
- Groupe JI + MI 1 (jeudi 14h) : Anne Micheli    anne.micheli@irif.fr
- Groupe BI + MI 2 (mercredi 14h) : Patrick Lambein-Monette  
lambein@irif.fr

toujours mentionner **[SY5]** dans le sujet du mail, et bien sûr, signer...

attention, il n'y a pas de machines en salle 2001 ; si possible, venir avec un portable ; sinon, m'écrire avec comme sujet **[SY5]** problème salle 2001

### Communication

principalement via [un dépôt git](#) (supports de cours, énoncés de TP, de projet...) :

`gaufre.informatique.univ-paris-diderot.fr/poulalho/sy5_2021-2022`

éventuellement moodle en solution de repli :

`moodle.u-paris.fr/course/view.php?id=1659`

probablement un serveur discord, mais il n'est pas encore mis en place

### Modalités de contrôle des connaissances

[Session 1](#) : 50% examen écrit + 50% projet en (bi- ou) trinômes

[Session 2](#) : examen écrit ou projet individuel

## PRÉREQUIS

langage C, en particulier pointeurs et allocation mémoire

shell (bash par exemple)

structures de données usuelles : tableaux, listes chaînées, tableaux dynamiques, tampon circulaire...

## QUEL EST LE RÔLE D'UN SYSTÈME D'EXPLOITATION ?

faire l'interface entre le matériel et l'utilisateur (humain ou applications)  $\implies$  fournir le bon niveau d'*abstraction*

## QUEL EST LE RÔLE D'UN SYSTÈME D'EXPLOITATION ?

faire l'interface entre le matériel et l'utilisateur (humain ou applications)  $\implies$  fournir le bon niveau d'*abstraction*

gérer les ressources, *équitablement et de façon sûre* : temps de calcul du processeur, mémoire, périphériques...

## QUEL EST LE RÔLE D'UN SYSTÈME D'EXPLOITATION ?

faire l'interface entre le matériel et l'utilisateur (humain ou applications)  $\implies$  fournir le bon niveau d'*abstraction*

gérer les ressources, *équitablement et de façon sûre* : temps de calcul du processeur, mémoire, périphériques...

en particulier, il effectue des opérations *critiques*; la plupart des ordinateurs ont deux modes opératoires : le mode **noyau**, où toutes les opérations sont permises, et le mode **utilisateur**, bridé.

## QUEL EST LE RÔLE D'UN SYSTÈME D'EXPLOITATION ?

faire l'interface entre le matériel et l'utilisateur (humain ou applications)  $\implies$  fournir le bon niveau d'*abstraction*

gérer les ressources, *équitablement et de façon sûre* : temps de calcul du processeur, mémoire, périphériques...

en particulier, il effectue des opérations *critiques*; la plupart des ordinateurs ont deux modes opératoires : le mode **noyau**, où toutes les opérations sont permises, et le mode **utilisateur**, bridé.

**But du cours** : comprendre (les grandes lignes de) la partie entre les deux traits, trait du haut inclus, trait du bas exclu, dans le cas des systèmes POSIX



## QUELQUES MOTS SUR LE MATÉRIEL

le processeur :

## QUELQUES MOTS SUR LE MATÉRIEL

le processeur :

- cycle de base : extraire une instruction – la décoder – l'exécuter

## QUELQUES MOTS SUR LE MATÉRIEL

le processeur :

- cycle de base : extraire une instruction – la décoder – l'exécuter
- ensemble d'instructions spécifique agissant sur la mémoire et les registres

## QUELQUES MOTS SUR LE MATÉRIEL

le processeur :

- cycle de base : extraire une instruction – la décoder – l'exécuter
- ensemble d'instructions spécifique agissant sur la mémoire et les registres
- parmi les registres : *compteur ordinal (program counter)*,  
*pointeur de pile (stack pointer)*

## QUELQUES MOTS SUR LE MATÉRIEL

le processeur :

- cycle de base : extraire une instruction – la décoder – l'exécuter
- ensemble d'instructions spécifique agissant sur la mémoire et les registres
- parmi les registres : *compteur ordinal* (*program counter*), *pointeur de pile* (*stack pointer*)
- changement de processus  $\implies$  sauvegarde des registres

## QUELQUES MOTS SUR LE MATÉRIEL

le processeur :

- cycle de base : extraire une instruction – la décoder – l'exécuter
- ensemble d'instructions spécifique agissant sur la mémoire et les registres
- parmi les registres : *compteur ordinal* (*program counter*), *pointeur de pile* (*stack pointer*)
- changement de processus  $\implies$  sauvegarde des registres

la mémoire : plusieurs niveaux, caractérisés par des vitesses et des capacités (très) différentes...

## QUELQUES MOTS SUR LE MATÉRIEL

le processeur :

- cycle de base : extraire une instruction – la décoder – l'exécuter
- ensemble d'instructions spécifique agissant sur la mémoire et les registres
- parmi les registres : *compteur ordinal (program counter)*, *pointeur de pile (stack pointer)*
- changement de processus  $\implies$  sauvegarde des registres

la mémoire : plusieurs niveaux, caractérisés par des vitesses et des capacités (très) différentes...

les périphériques d'entrée/sortie : en général, contrôleur + périphérique lui-même  
nécessitent des *pilotes (device drivers)*

## QUELQUES CONCEPTS IMPORTANTS

les **processus** : objet dynamique représentant un programme en cours d'exécution



## QUELQUES CONCEPTS IMPORTANTS

les **processus** : objet dynamique représentant un programme en cours d'exécution

- *espace d'adressage* contenant le programme exécutable, ses données, sa pile

## QUELQUES CONCEPTS IMPORTANTS

les **processus** : objet dynamique représentant un programme en cours d'exécution

- *espace d'adressage* contenant le programme exécutable, ses données, sa pile
- registres... à sauvegarder en cas d'interruption

## QUELQUES CONCEPTS IMPORTANTS

les **processus** : objet dynamique représentant un programme en cours d'exécution

- *espace d'adressage* contenant le programme exécutable, ses données, sa pile
- registres... à sauvegarder en cas d'interruption
- $\Rightarrow$  **table des processus**

## QUELQUES CONCEPTS IMPORTANTS

les **processus** : objet dynamique représentant un programme en cours d'exécution

- *espace d'adressage* contenant le programme exécutable, ses données, sa pile
- registres... à sauvegarder en cas d'interruption
- $\implies$  **table des processus**

**mode utilisateur** *vs* **mode noyau** : actions inoffensives *vs* actions sensibles  $\implies$  **appel système** pour passer de l'un à l'autre

## QUELQUES CONCEPTS IMPORTANTS

mode utilisateur *vs* mode noyau : actions inoffensives *vs* actions sensibles  $\implies$  appel système pour passer de l'un à l'autre

## QUELQUES CONCEPTS IMPORTANTS

mode utilisateur *vs* mode noyau : actions inoffensives *vs* actions sensibles  $\implies$  appel système pour passer de l'un à l'autre

appel système : syntaxiquement semblable à un appel de fonction, mais beaucoup plus lent à cause des changements de contexte

## QUELQUES CONCEPTS IMPORTANTS

`mode utilisateur` *vs* `mode noyau` : actions inoffensives *vs* actions sensibles  $\implies$  `appel système` pour passer de l'un à l'autre

`appel système` : syntaxiquement semblable à un appel de fonction, mais beaucoup plus lent à cause des changements de contexte

documentés dans la section 2 du manuel (section 3 pour les fonctions C usuelles)

## QUELQUES CONCEPTS IMPORTANTS

`mode utilisateur` *vs* `mode noyau` : actions inoffensives *vs* actions sensibles  $\implies$  `appel système` pour passer de l'un à l'autre

`appel système` : syntaxiquement semblable à un appel de fonction, mais beaucoup plus lent à cause des changements de contexte

documentés dans la section 2 du manuel (section 3 pour les fonctions C usuelles)

Sous linux, on peut voir la liste des appels système effectués par un processus grâce à la commande `strace`.



## QUELQUES CONCEPTS IMPORTANTS

les **fichiers** : abstraction qui masque les particularités des disques de stockage et autres périphériques d'entrée/sortie

## QUELQUES CONCEPTS IMPORTANTS

les **fichiers** : abstraction qui masque les particularités des disques de stockage et autres périphériques d'entrée/sortie

**appels système nécessaires** : pour la création, la destruction, les accès en lecture et écriture

## QUELQUES CONCEPTS IMPORTANTS

**les fichiers** : abstraction qui masque les particularités des disques de stockage et autres périphériques d'entrée/sortie

**appels système nécessaires** : pour la création, la destruction, les accès en lecture et écriture

dans la plupart des systèmes, structuration de l'ensemble des fichiers grâce à la notion de **répertoires**, qui permet d'établir une *hiérarchie*

## APPELS SYSTÈME POUR LES ENTRÉES/SORTIES

l'accès à un fichier est une action critique  $\implies$  appel système

```
int open(const char *pathname, int flags);  
int open(const char *pathname, int flags, mode_t mode);
```

renvoie un entier appelé **descripteur de fichier**, ou -1 en cas d'échec

## APPELS SYSTÈME POUR LES ENTRÉES/SORTIES

l'accès à un fichier est une action critique  $\Rightarrow$  appel système

```
int open(const char *pathname, int flags);  
int open(const char *pathname, int flags, mode_t mode);
```

renvoie un entier appelé *descripteur de fichier*, ou -1 en cas d'échec

- *table des descripteurs du processus*
- *table des fichiers ouverts du système*

## APPELS SYSTÈME POUR LES ENTRÉES/SORTIES

l'accès à un fichier est une action critique  $\implies$  appel système

```
int open(const char *pathname, int flags);  
int open(const char *pathname, int flags, mode_t mode);
```

renvoie un entier appelé **descripteur de fichier**, ou -1 en cas d'échec

- **table des descripteurs** *du processus*
- **table des fichiers ouverts** *du système*

vous connaissez déjà des descripteurs :

- 0 est le descripteur associé à l'entrée standard ;
- 1 est le descripteur associé à la sortie standard ;
- 2 est le descripteur associé à la sortie erreur standard.

note : ces trois descripteurs sont aussi définis par des macros dans `unistd.h` : `STDIN_FILENO`, `STDOUT_FILENO` et `STDERR_FILENO`.

## APPELS SYSTÈME POUR LES ENTRÉES/SORTIES

```
ssize_t read(int fd, void *buf, size_t count);  
ssize_t write(int fd, const void *buf, size_t count);
```

(les types `size_t` et `ssize_t` sont des entiers respectivement non signés et signés pour POSIX.1. Ils servent essentiellement à conserver la compatibilité entre les différentes versions de POSIX)

- `fd` est un descripteur
- `count` est la taille des données à lire ou écrire
- `buf` est l'adresse d'un emplacement mémoire pour stocker les données lues ou lire les données à écrire