Fiche de révision enrichie – Système d'Exploitation Linux (niveau EPITA/partiels)

I. Notions fondamentales

Programme vs Processus

- **Programme** :
    - *Objet passif* : code source ou binaire, fichier sur disque.
- **Processus** :
    - *Objet actif* : instance d'un programme en cours d'exécution.
    - Plusieurs processus peuvent exécuter *simultanément* le même programme (ex : ouvrir 2 fois Firefox).
    - Un même processus peut *enchaîner* plusieurs programmes successivement via `exec` (ex : un shell qui lance `ls`).

II. Segments d'un Programme

- Un programme en mémoire est divisé en **segments** :
    - **.text** : instructions (code exécutable)
    - **.data** : données initialisées en écriture/lecture
    - **.bss** : données non-initialisées (remplies de 0 à l'allocation)
    - **.rodata** : données initialisées, en lecture seule (ex : chaînes de caractères constantes)
- **Droits possibles sur un segment** : lecture (r), écriture (w), exécution (x)

III. Modes d'exécution

- **Kernel mode (privilégié, Ring 0)** :
    - Le noyau peut exécuter toutes les instructions du processeur.
    - Accès complet à toute la mémoire et au matériel.
    - Peut accéder à l'espace mémoire du kernel *et* des processus utilisateur.
- **User mode (non privilégié, Ring 3)** :
    - Processus *n'a pas accès* à la mémoire du noyau.
    - Accès restreint : interdit certaines instructions sensibles (I/O direct, gestion mémoire...).
- **Passage user → kernel** :
    - Se fait par *appel système* (`syscall`), jamais par appel de fonction standard C (`malloc`, etc).
    - Retourne en user mode à la fin du syscall.

IV. Hiérarchie mémoire

- Classée par **vitesse croissante** : registres < cache < RAM < disque.
- Classée par **capacité croissante** : registres < cache < RAM < disque.
- **Stockage primaire** : directement accessible par le CPU (RAM).
- **Stockage secondaire** : non-volatile, accès indirect via OS (HDD, SSD).

V. Gestion mémoire virtuelle

- **Rôles** :
    - *Isolation/protection* des processus : pas de fuite d'information/mauvais accès entre processus.
    - *Découplage* entre adresses virtuelles et physiques, simplifie chargement/programmes plus facilement.
- **Mécanismes** :
    - Pagination (pages/frames)
    - Table des pages (Page Table) : mapping virtuel→physique, contient présence, permissions, validité.
    - Attributs possibles : page présente ou non, accessible en user mode, permissions…
    - **Traduction accélérée par TLB** (Translation Lookaside Buffer) : cache les mappings récents.
- **Appels système gestion mémoire** :
    - `mmap()` : mappe des pages dans l'espace virtuel d'un processus.
    - `munmap()` : démappe des pages.
    - *Attention* : `malloc`, `realloc`, `free` sont des fonctions C qui utilisent éventuellement `mmap`/`brk` derrière.

VI. Pagination 32 bits (1 niveau)

- **Offset** : 12 bits (pour 4096 octets/page)
- **Entrées table des pages** : 2^{20} (1M entrées)
- **Taille d'une page/frame** : 4096 octets (2^{12})
- **Taille table des pages** : 4 Mo (2^{22} octets si chaque entrée = 4 octets)
- **Attributs dans une entrée** : présence, permissions, accès user/kernel…

VII. Faute de page (Page Fault)

- Causes principales :
    - La page n'appartient pas à l'espace d'adressage virtuel du processus.
    - La page n'est pas présente en mémoire centrale (swap ou pas encore allouée).

VIII. ASLR (Address Space Layout Randomization)

- *L = Layout*
- Technique de sécurité : randomise les adresses virtuelles (pas physiques) des segments (heap, stack, libs…).
- But : compliquer l'exploitation des failles mémoire.

IX. Commandes systèmes à connaître

- **strace** : liste les appels système d'un processus
- **vmstat** : affiche stats mémoire et système
- **readelf** : analyse la structure ELF (pas la table des syscalls !)
- **free** : affiche mémoire libre/occupée (ne libère rien !)

X. Signaux & gestion

- **Signal** : notification asynchrone envoyée à un processus (ex : SIGKILL, SIGTERM, SIGINT…)
- **Actions par défaut** :
    - *Ignorer*
    - *Terminer* le processus
    - *Stopper* ou *reprendre* l'exécution
    - *Exception* : handler custom (écrit par le programme)
- **SIGKILL** :
    - Ne peut *pas* être intercepté ni ignoré
    - Force l'arrêt du processus sans appel de son code
    - Différent de SIGTERM qui peut être attrapé/intercepté

XI. Autres notions à ne pas rater

- **Ordonnancement** : FCFS, Round-Robin, Priorités, CFS.
- **Processus/Zombie** : processus terminé mais pas encore collecté par le parent.
- **ELF** : format binaire standard Linux, sections à connaître (.text, .data, .bss, .rodata).
- **Différences malloc/calloc** : `calloc` alloue ET initialise à 0, `malloc` n'initialise pas.
- **Copy-on-write (fork)** : pages dupliquées uniquement si modification.