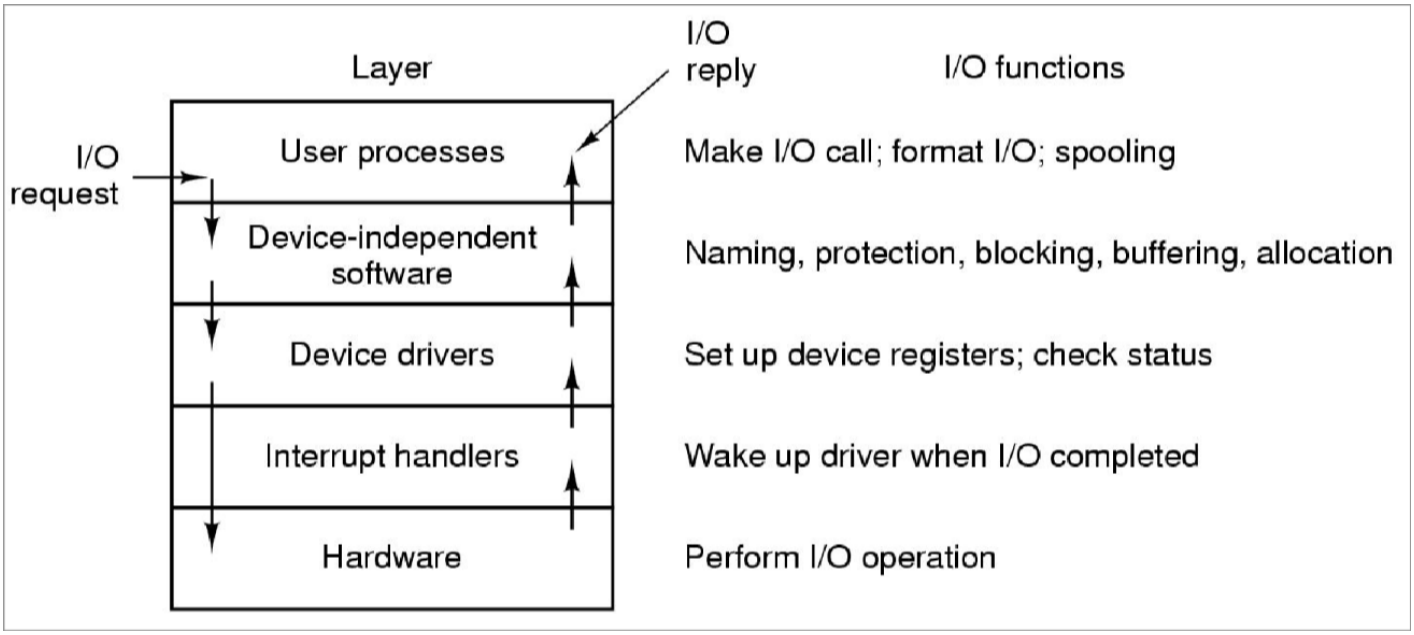
Côté logiciel

La partie logicielle de l'entrée-sortie est une partie du système d'exploitation qui a pour but de fournir une interface vers les périphériques tout en assurant une indépendance par rapport au type de périphérique.

Par exemple, l'accès à une clé USB doit être, du point de vue des programmes, identique à l'accès sur un disque dur.

Cette partie du système d'exploitation est divisée en quatre couches, que l'on va adresser en partant du plus proche du matériel vers le plus haut niveau.



Le gestionnaire d'interruption

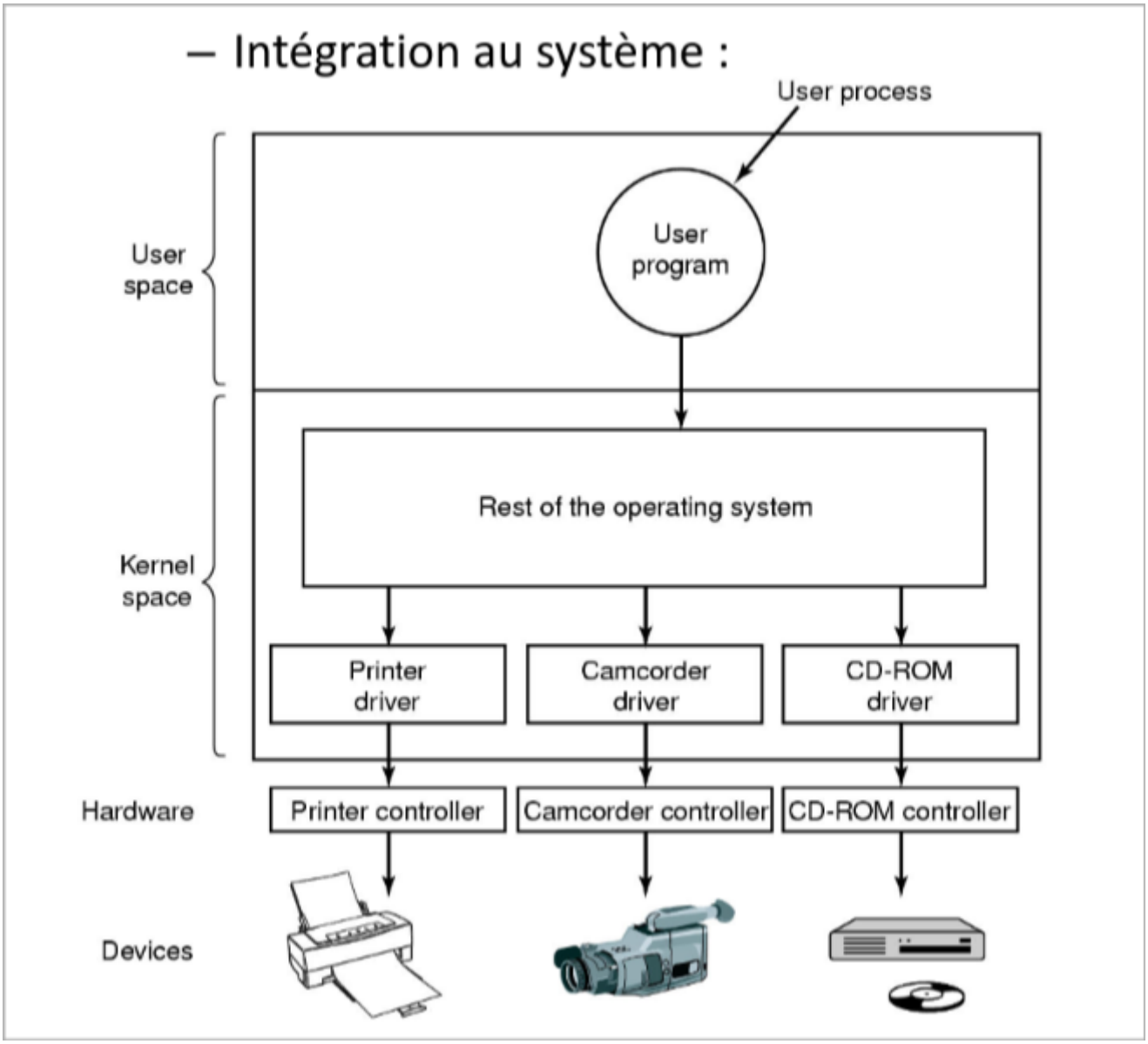
Le gestionnaire de périphérique (le pilote) qui lance l'opération d'entrée-sortie, va attendre une interruption.

Lorsque l'interruption survient, **gestionnaire d'interruptions** va avertir le gestionnaire de périphérique et celui-ci va terminer l'opération voulue.

Lorsqu'une interruption survient, le gestionnaire d'interruptions va sauvegarder le contexte du processus en cours, créer le contexte pour le traitement de l'interruption, se place en état "disponible" pour traiter les interruptions suivantes, exécuter la procédure de traitement (la routine dont on a parlé dans la section précédente).

Une fois les interruptions gérées, le scheduler choisit le processus à démarrer.

Gestionnaire de périphérique (pilote ou driver)



Le **gestionnaire de périphérique**, aussi appelé **pilote** ou **driver**, dépends de la nature du périphérique, ainsi un gestionnaire de disque fonctionne très différemment d'un gestionnaire de souris.

Le gestionnaire de périphérique, afin d'être utilisable, doit être chargé au cœur du système, dans le noyau, en mode protégé. C'est pour cela qu'un mauvais driver pour conduire à des plantages du système d'exploitation.

Mais en même temps, ces gestionnaires de périphériques, écrits par les fabricants, doivent pouvoir être inséré facilement dans le noyau (kernel) pour être utilisé.

Fonctionnement

Le système appelle des fonctions internes (open, read, write, initialize, etc) des pilotes afin de commander le périphérique. Les pilotes vont ensuite vérifier si les paramètres reçus sont corrects et valide et vont traduire certains paramètres en données physiques.

Par exemple, un pilote de disque dur va traduire un numéro de bloc en cylindre, piste, secteur et tête.

Ensuite, le pilote vérifie si le périphérique est disponible (sinon il attend), il vérifie également l'état du périphérique et le commande.

Pendant le travail du périphérique, le pilote s'endort, car il ne peut plus rien faire. Une fois qu'il est réveillé par le gestionnaire d'interruption, il vérifie que tout s'est bien passé et transmet les informations.

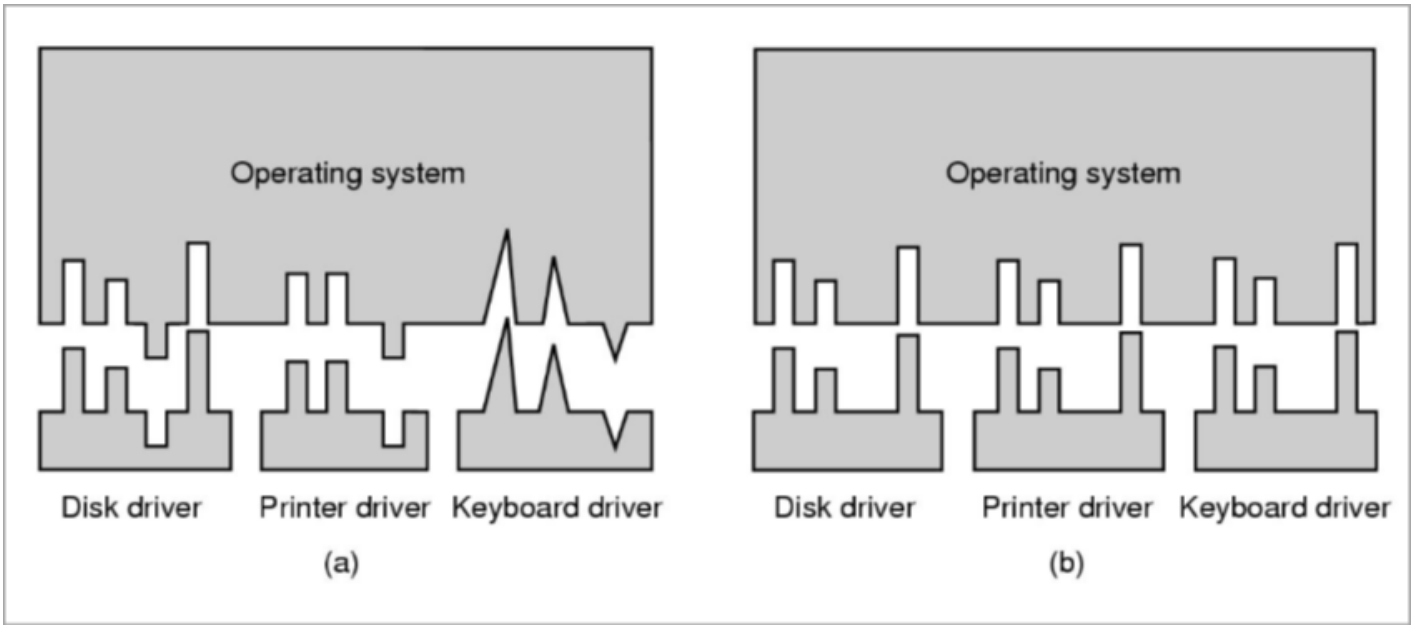
La couche logicielle indépendante

La couche de **logicielle indépendante** (aussi appelée **interface standard**)

Cette couche fournit une interface uniforme pour les drivers (la fameuse API dont on a parlé plus tôt), ainsi que du buffering, de la gestion d'erreur, de l'allocation et libération des périphériques.

Interface standard

Grâce à cette couche, on s'assure que les drivers sont appelés de manière uniforme.



A gauche, un système sans interface standard, à droite un système avec une interface standard

Uniformisation des noms

La désignation d'un périphérique doit être non ambiguë, sous Linux le répertoire `/dev` contient des entrées pour chaque périphérique. Un périphérique est donc traité au même titre qu'un fichier, il ne peut donc pas en avoir deux avec le même nom.

Par exemple, un disque dur sera appelé `/dev/sda`, si on met un autre disque dur, celui-ci sera appelé `/dev/sdb`.

Buffering

Afin d'améliorer les performances du système et d'éviter de faire beaucoup d'accès inutilement, on garde des buffers (tampons) dans le système d'exploitation et dans les contrôleurs. L'objectif étant de placer en mémoire les informations qui sont souvent accédées.

Gestion d'erreurs

C'est aussi à cette couche que revient la gestion des erreurs. L'utilisation de périphériques induit un certain nombre d'erreurs (temporaire ou permanentes).

Il y a deux types d'erreurs, les **erreurs de programmation** qui surviennent lorsque l'opération demandée est impossible (exemple, écriture sur un périphérique d'entrée), dans quel cas on rapporte l'erreur et on s'arrête.

Et les **erreurs d'entrées sorties** qui surviennent lorsqu'une opération se termine de façon anormale (par exemple, on déconnecte le disque, ou le périphérique est défectueux). Dans ce cas, le driver décide de soit tenter à nouveau l'opération ou alors de rapporter l'erreur.

Allocations et libérations des périphériques

Certains périphériques ne sont pas partageables, par exemple avec une imprimante, il est impossible d'imprimer plusieurs choses en même temps.

Il revient alors au système d'exploitation de vérifier si le périphérique est libre et s'il peut être alloué au processus.

La fonction `open` permet à un processus d'informer le système d'exploitation qu'il souhaite utiliser un périphérique, le système vérifie alors sa disponibilité.

Couche d'entrée-sortie applicative

Cette partie de la gestion de l'entrée sortie est faite au niveau de l'application, par exemple via les libraires systèmes liées aux programmes.

C'est là que l'on va par exemple trouver les appels systèmes de `stdio` tel qu'open, printf, read, write, etc.

Cette couche va également fournir un système de **spooling** qui permet de créer une fille d'attente pour l'accès aux périphériques non partageables. Par exemple via une liste de jobs d'impression pour une imprimante.

Revision #1
Created 5 January 2024 13:27:42 by SnowCode
Updated 6 January 2024 19:13:15 by SnowCode