ensiie
école nationale supérieure d'informatique
pour l'industrie et l'entreprise

Systèmes Informatiques
APPELS NOYAU

G.BERTHELOT

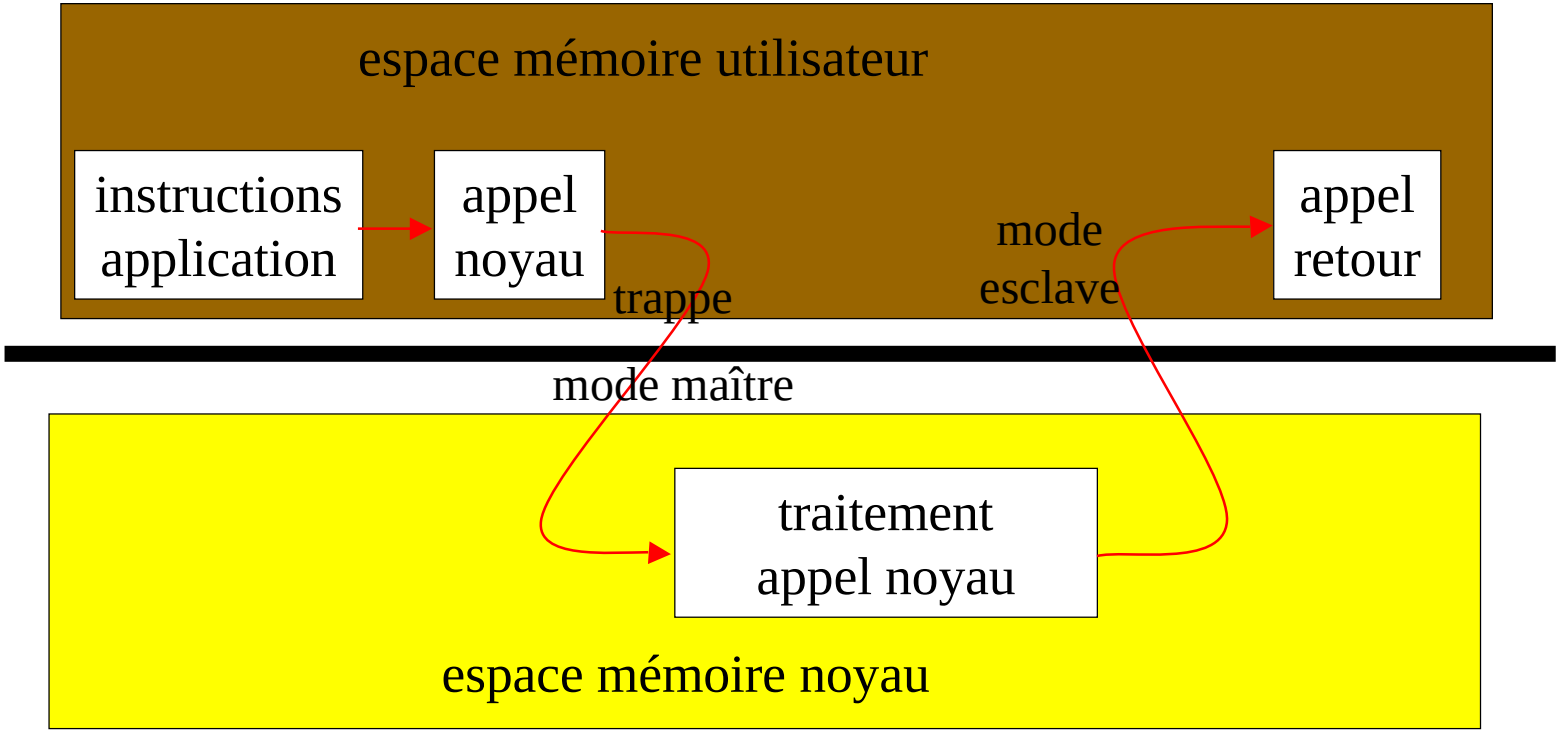
Noyau unix

- Noyau : collection d'appels système sur
  - les fichiers,
  - les processus,
  - les entrées/sorties,
  - communications inter-processus,
  - communications réseaux,
  - etc.

  (documentés dans la section 2 du man)
- Chaque appel se présente comme un appel de fonction.
- La plupart des appels noyau retournent une valeur qui indique le succès ou l'échec de l'opération demandée. En cas d'échec la cause exacte de l'échec peut être précisée par le contenu de la variable globale errno disponible en incluant le fichier <errno.h>. Celui-ci contient également la liste des codes d'erreur.

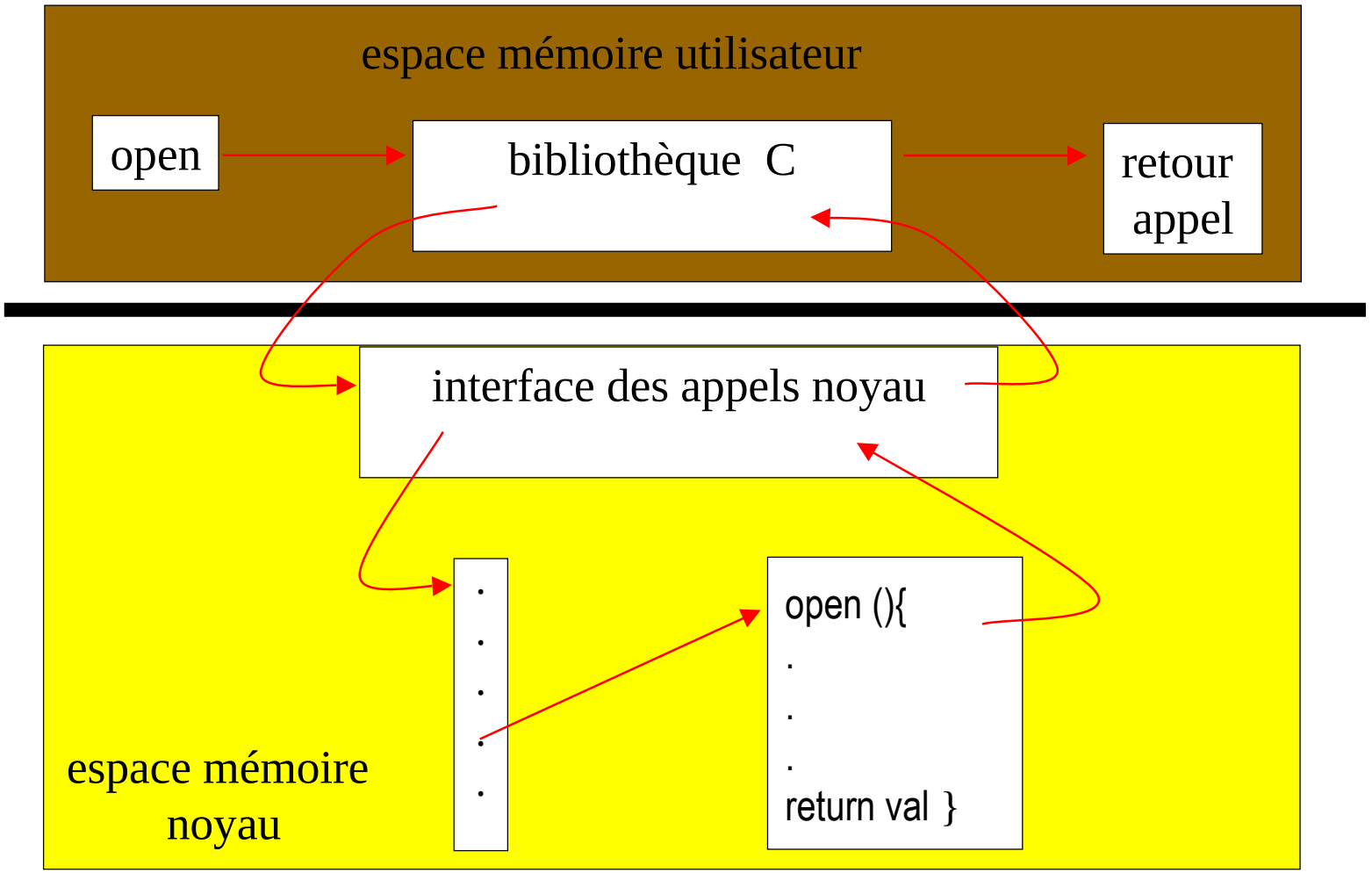
Noyau unix

- changement espace mémoire et mode exec.



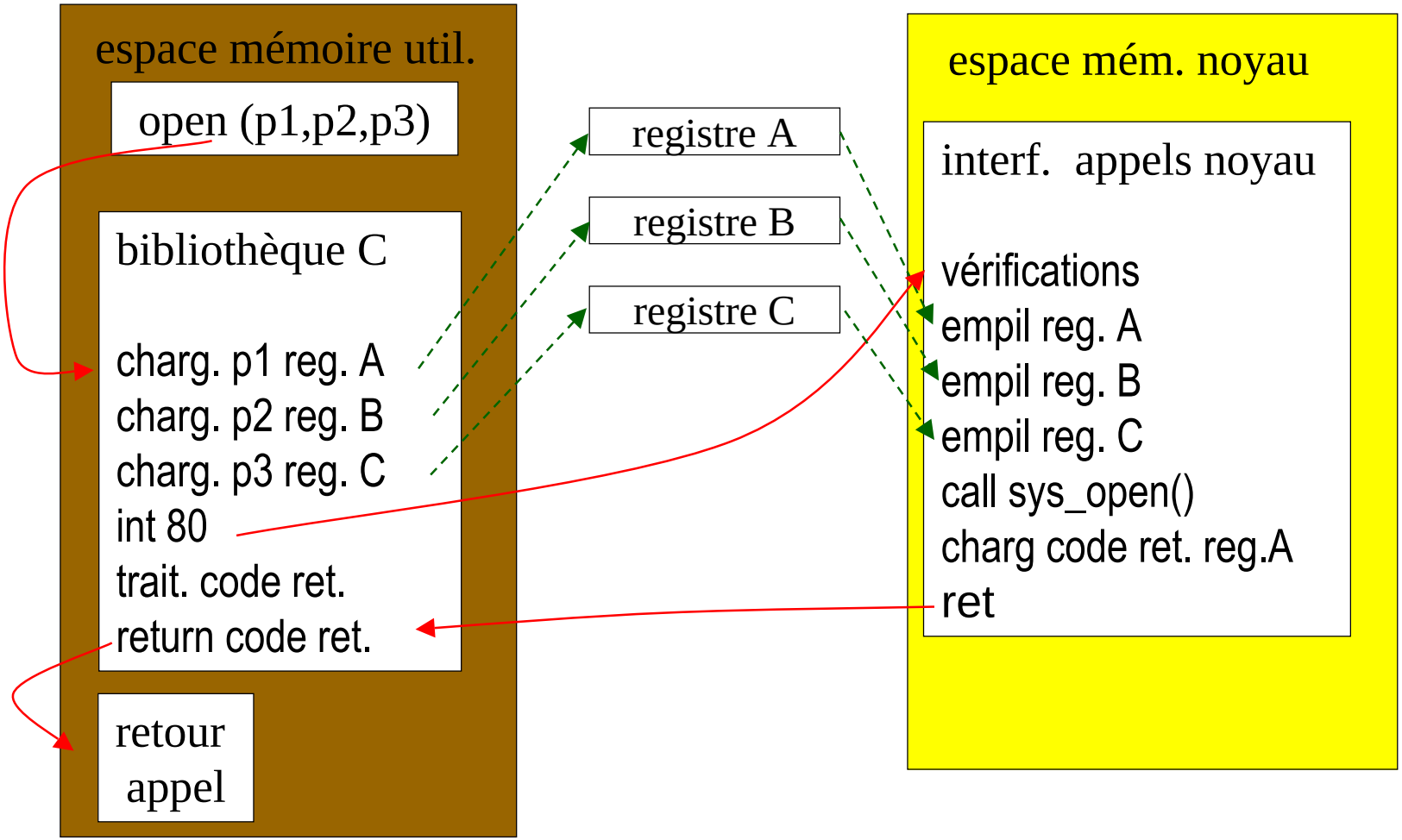
Noyau unix

- passage par la bibliothèque C



Noyau unix

- passage des paramètres



Noyau unix

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- L'appel noyau est exécuté par le processus lui-même mais:
  - les fonctions système sont exécutés en utilisant une pile d'exécution différente (une par processus)
  - la priorité d'exécution est plus grande ;
  - certaines interruptions peuvent être masquées (par exemple pendant les commutations)

  Ceci évite de faire une commutation et simplifie beaucoup le passage de paramètres et de valeur de retour puisqu'on reste dans le même espace d'adresses.

- idem pour le traitement des interruptions, même si elles n'ont pas de rapport avec le processus en cours

Fichiers unix

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- Tout fichier est défini par un descripteur de fichier unique appelé i-noeud (inode).
- contient les informations de gestion associées fichier :
  - taille en nombre d'octets;
  - adresse des blocs utilisés sur le disque ;
  - identification du propriétaire ;
  - permissions d'accès ;
  - type de fichier (fichier ordinaire, catalogue, ...) ;
  - date de dernière modification ;
  - compteur de références à ce fichier dans un répertoire.
- L'i-noeud ne contient aucun nom pour le fichier.
- Un fichier est détruit lorsque le compteur de réf. vaut 0

Fichiers unix

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- Un ficher régulier UNIX peut être considéré comme un tableau de caractères et les opérations de lecture/écriture se font à partir d'un "pointeur de position" qui pointe sur le premier caractère lors de l'ouverture et est ensuite avancé au fur et à mesure des lectures et des écritures.

Fichiers unix

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

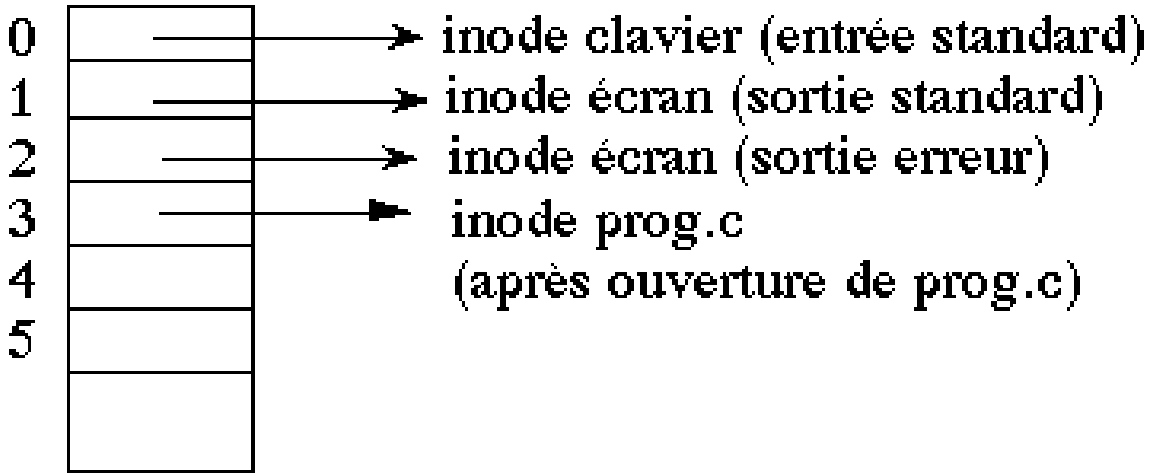
- En plus des permissions d'accès, 2 autres bits sont utilisables pour chaque fichier :
  - set-uid permet d'exécuter un programme avec les privilèges de son propriétaire et non pas ceux de l'utilisateur qui lance l'exécution.

  Ce mécanisme est utilisé en particulier pour changer le mot de passe (/sbin/passwd)

  - set-gid : même chose avec le groupe
- Le bit set-uid à 1 se traduit par une lettre s dans les permissions d'accès (affichées lors d'un ls -l) ;

Fichiers unix

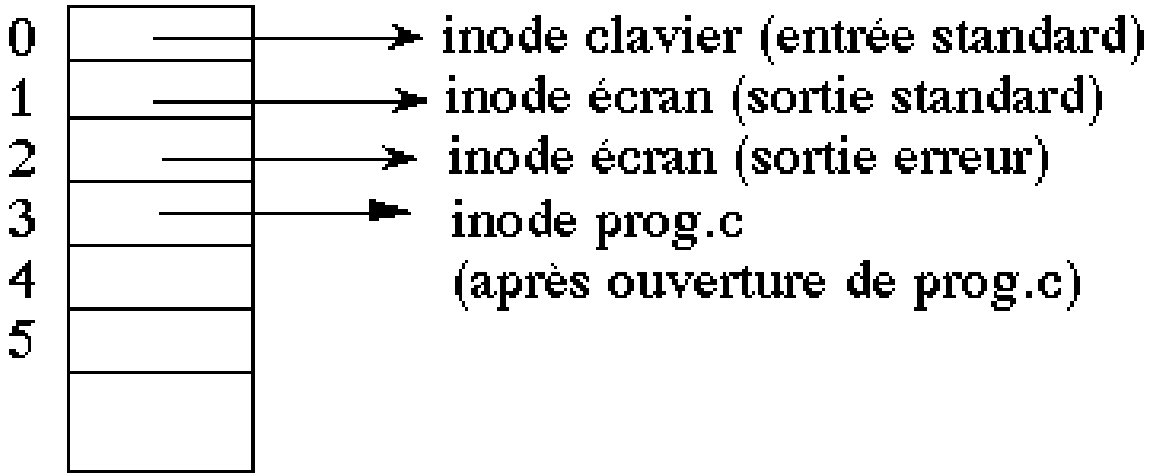
- permet à un processus d'accèder aux divers flux (fichiers, tubes, socket) après leur ouverture



Elle est modifiée à chaque ouverture et à chaque fermeture

Fichiers unix

- permet à un processus d'accèder aux divers flux (fichiers, tubes, socket) après leur ouverture



Elle est modifiée à chaque ouverture et à chaque fermeture

Fichiers unix

- Toute utilisation d'un fichier commence par une ouverture qui renvoie un "*file descriptor*" en abrégé *fd* traduit ici par "numéro de fichier ouvert" :
- *fd* : numéro de l'entrée qui contient un pointeur vers l'i-noeud du fichier, dans la table des fichiers ouverts.

- Avantage : inutile de retraduire le nom à chaque opération sur le fichier.

Fichiers unix

- int open(char * référence, int mode, int permis)

- Si le fichier désigné par la référence relative ou absolue existe alors il est ouvert et permis peut être omis.
- mode indique le type d'accès : O_RDONLY,O_WRONLY et O_RDWR pour respectivement : lecture seulement, écriture seulement et lecture/écriture.
- La fonction open retourne le numéro de fichier ouvert en cas de succès et -1 sinon.

Fichiers unix

- Le pointeur de position du fichier pointe sur le premier octet du fichier.
- Exemple :
- int nf;
- nf = open("projet.pas", O_RDWR);

Fichiers

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- int close (int no_fichier);
- Ferme le fichier de numéro no_fichier.
- L'entrée dans la table des fichiers ouverts est libérée et peut être réutilisée ultérieurement lors de l'ouverture d'un autre fichier.
- La fonction close retourne 0 si l'opération se termine normalement et -1 sinon.

Fichiers

- Exemple :
- int nf, ff;
- nf = open("projet.pas", O_RDWR);
- ff = close(nf);

Fichiers

- int read(int no_fichier, char *adr_zone, int nb);

- Essaye de lire nb octets à partir de la position indiquée par le pointeur de position du fichier désigné par no_fichier et les copie à l'emplacement désigné par adr_zone.
- La valeur retournée est le minimum de nb et de la quantité effectivement disponible et est égale à zéro si la fin de fichier est atteinte.
- Le pointeur de position du fichier est augmenté de la quantité lue.

Fichiers

- Exemple :

```
int nb;
char c, tab[100];
nb = read(0,&c,1);
nb = read(nf, tab, sizeof(tab));
```

Fichiers

- int write(int no_fichier, char *adr_zone, int nb);

- Transfère nb caractères depuis l'adresse désignée par adr_zone à l'endroit pointé par le pointeur de position du fichier no_fichier.

- Retourne le nombre de caractères transférés dans le fichier et -1 en cas de problème.

Fichiers

- Exemple :
- int nb;
- char c, tab[8]="bonjour";
- nb = write(1,&c,1);
- nb = write(nf,tab,sizeof tab);

Fichiers

- long lseek(int no_fichier, long déplacement, int drapeau);

- Déplace le pointeur de position du fichier désigné par no_fichier de déplacement octets par rapport à la position courante, par rapport au début du fichier, ou par rapport à la fin du fichier suivant que drapeau est égal respectivement à 0 (ou SEEK_SET), 1 (ou SEEK_CUR), 2 (ou SEEK_END).
- Retourne la valeur du pointeur de position après l'opération et -1 en cas d'erreur.
- Le début du fichier correspond à la valeur 0.

Fichiers

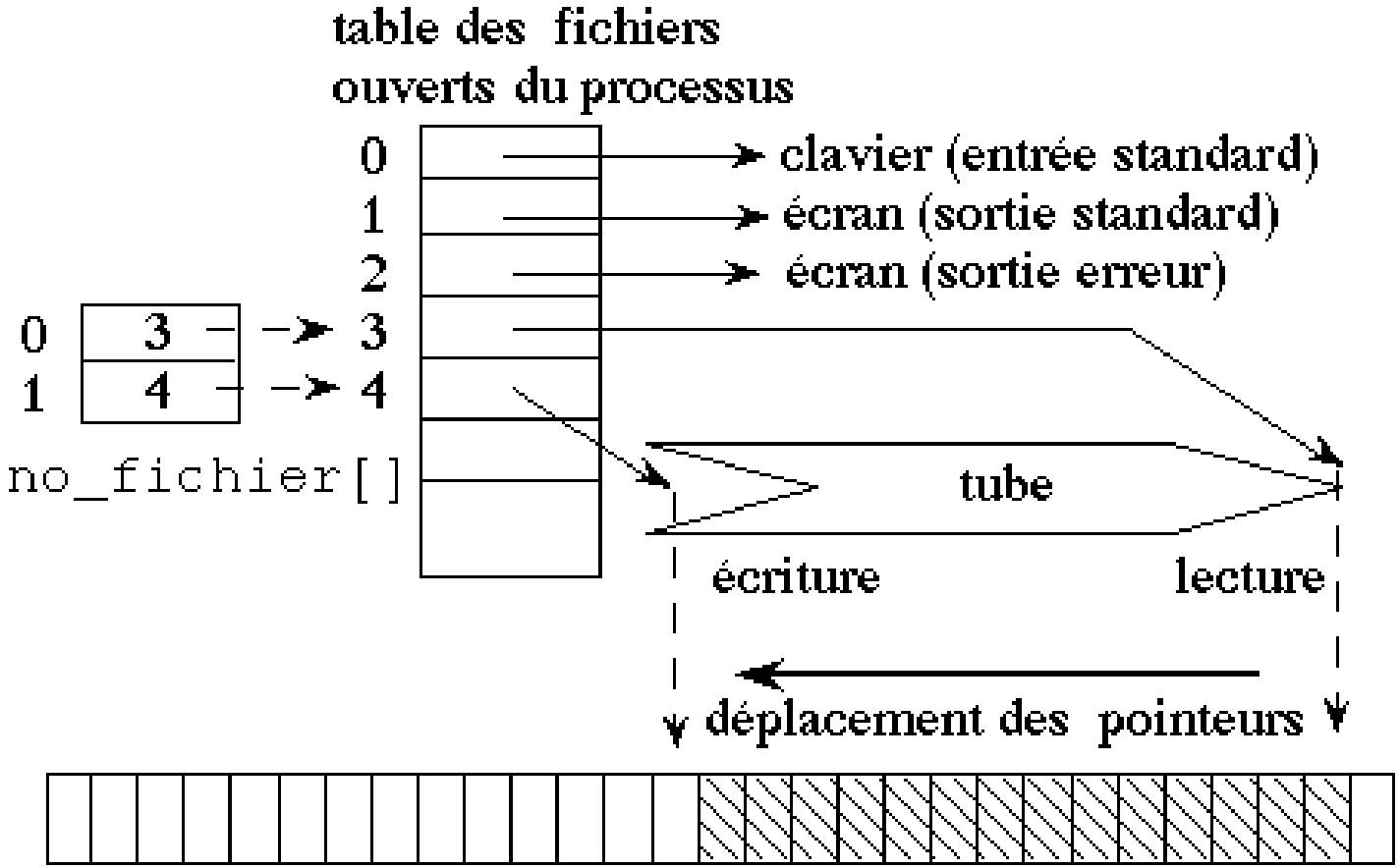
- creat : création d'un fichier.
- link : création d'un lien physique sur un fichier.
- unlink : suppression d'un lien physique sur un fichier et suppression du fichier si c'était le dernier lien existant.
- dup : duplication d'une entrée dans la table des fichiers ouverts dans la première entrée disponible.
- chdir : changement de répertoire de travail.
- chown : changement de propriétaire de fichier.
- chmod : changement de permission d'accès.

Fichiers

- flush : vide un tampon d'entrée/sortie.
- stat : donne les attributs d'un fichier (type, propriétaire, permission d'accès, etc).
- umask : modification du "masque" des permissions d'accès utilisé lors des créations de fichiers.

Fichiers

- tubes (pipes):



Fichiers

- int pipe (int no_fichier[2]);

- créé et ouvre le tube à la fois en lecture et en écriture et rend deux numéros de fichiers ouverts, l'un correspondant à l'ouverture en lecture et l'autre à l'ouverture en écriture.

- Retourne 0 si la création du tube a bien eu lieu et -1 sinon.

Fichiers

- Exemple :
- int retour, no_fichier[2];
- retour = pipe(no_fichier);

- Un tube ne peut être utilisé que par le processus qui l'a créé ou par ses descendants (du fait de son absence de nom).

- L'écriture et la lecture dans un tube se font en utilisant les appels normaux read() et write() avec comme paramètre le numéro de fichier correspondant.

Fichiers

- Une "extrémité" d'un tube peut être fermée par appel de close() sur le numéro de fichier ouvert correspondant.
- Un tube fermé à une extrémité par un processus ne peut être ré-ouvert par ce même processus (absence de nom de tube).
- Un tube est définitivement fermé en écriture lorsque tous les processus qui possédaient le numéro de fichier ouvert en écriture l'ont fermé en écriture.
- Un tube est définitivement fermé en lecture lorsque tous les processus qui possédaient le numéro de fichier ouvert en lecture l'ont fermé en lecture.

appels noyau sur les fichiers

- Écrire dans un tube définitivement fermé en lecture n'a pas de sens.

  Toute écriture dans un tube définitivement fermé en lecture provoque l'envoi d'un signal "tube fermé" (SIGPIPE) au processus demandeur.

  Sauf mention contraire ce signal provoque sa mort (voir signaux).

Processus

- Un processus est une exécution d'un programme à un instant donné. Le programme lui-même est un objet inerte rangé sur disque sous forme d'un fichier ordinaire exécutable. Plusieurs caractéristiques sont associées à un processus, dont :
  - une identification ou Process IDentifier ou PID (entier) ;
  - un propriétaire réel ou Real User IDentifier (entier);
  - un propriétaire effectif ou Effective User IDentifier (entier);
  - un groupe propriétaire réel ou Real Group IDentfifier (entier) ;
  - un groupe propriétaire effectif ou Effective Group ID (entier)
  - un terminal d'attachement (le terminal du login).
- Lors du lancement d'un processus, le programme qu'il doit exécuter est chargé en mémoire centrale en vue de son exécution.

Processus

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- Différents types de processus:
- Un processus système est un processus créé lors du lancement du système pour exécuter des tâches à l'intérieur du noyau :
  - Gestion des pages de mémoire,
  - Transferts des processus suspendus sur disque ...
  - créé lors du démarrage
- Un démon est un processus chargé d'une tâche d'intérêt général :
  - Gestion d'impression, gestion réseau, gestion du courrier ...
  - Il n'est pas attaché à un terminal.
  - créé au démarrage ou par l'administrateur
  - interrompu par l'administrateur ou lors de l'arrêt du système.

Processus

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- Différents types de processus:

- Un processus utilisateur est lancé par un utilisateur particulier depuis un terminal donné.
- Lors du login sur un terminal, un processus est lancé (shell en général, mais ce peut être un autre programme, indiqué par le fichier /etc/passwd).

Processus

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- Création de processus
  - Création par clonage d'un processus père qui utilise l'appel système fork().
  - Le processus fils est une copie exacte du processus père avec :
    - le même programme mais
    - des copies des données du père.
  - En cas de succès l'appel noyau
    - fork renvoie 0 au processus fils,
    - renvoie le numéro du processus fils au processus père.

Processus

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- Création de Processus:
- int fork();

  Crée un processus, le "fils", copie du processus appelant, le "père".
- Le processus fils :
  - exécute le même programme que le processus père;
  - utilise des copies des données du père;
  - mêmes fichiers ouverts que le père et mêmes pointeurs de position;
  - mêmes propriétaires réel et effectif que le père;
  - mêmes réactions aux signaux que le processus père;
  - même répertoire de travail que le père;
  - attaché au même terminal que le père;
  - identificateur de processus (PID) différent de celui du père.

Processus

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- Exécution du processus fils
  - Le processus fils commence son exécution en terminant la fonction fork

  => un seul processus (le père) commence l'exécution de fork,

  => mais deux processus (le père et le fils) terminent l'exécution de fork.

Processus

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

```
main (){
int pid;
printf("au nom du père\n"); /* 1 */
pid = fork(); /* 2 */
if (pid == 0) /* 3 */
printf("au nom du fils\n"); /* 4 */
if (pid != 0) /* 5 */
printf("au nom du père\n"); /* 6 */
printf("au nom du père et du fils\n"); /* 7 */
}
```

Processus

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- exit (int n);

- Termine le processus qui l'appelle. La valeur de n est disponible pour le processus père du processus appelant et il peut l'obtenir par la fonction wait (voir plus loin).

- La fonction exit provoque la fermeture de tous les fichiers ouverts par le processus et la libération de toutes les ressources détenues par le processus (mémoire, etc).

Processus

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- Les processus fils du processus qui se termine sont orphelins mais sont adoptés par le processus "init" (processus1).
- L'entrée correspondant au processus dans la table des processus est libérée.
- L'identificateur de processus ne sera pas réutilisé avant un certain temps pour éviter des confusions.
- Il n'y a pas de retour pour cet appel système.

Processus

- int wait(int *n);
- Permet de suspendre un processus jusqu'à la réception d'un signal ou qu'un de ses processus fils se termine ou s'arrête.
- Renvoie le numéro du processus qui s'est terminé.
- Si n est un pointeur non nul, la valeur de l'entier *n contient des informations sur l'arrêt du processus fils (nature de la fin du fils dans les 8 bits de poids faible et valeur d'appel du exit() par le fils dans les bits de poids fort, voir man pour les macros).
- S'il n'y a pas de processus fils, la primitive wait retourne immédiatement la valeur -1.

Processus

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- execlp(char *ref, char *arg0, ..., char *argn,0).

- le texte du nouveau programme recouvre et efface celui de l'ancien programme exécuté.
- => pas de retour d'un exec réussi. Les appels à ces fonctions n'entraînent pas de création de processus mais seulement une modification de l'image exécutée.

- Permet de lancer l'exécution du fichier dont la référence est ref avec les paramètres arg1,..., argn. Le paramètre arg0 est obligatoire et doit être la même chaîne de caractères que ref.

Processus

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- execlp(char *ref, char *arg0, ..., char *argn,0).

- La table des fichiers ouverts ne change pas : les fichiers ouverts avant l'exec restent ouverts
- Les traitements des signaux sont perdus.

Processus

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- sleep(int n) : met le processus en sommeil pour n secondes.
- getpid() : fournit le numéro du processus.
- getppid() : fournit le numéro du processus père.
- getuid() : donne le propriétaire.
- getgid() : donne le groupe du processus.
- setuid(int id) : change le propriétaire.
- setgid(int gid) : change le groupe du processus.

Signaux

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- Rôle : Avertir un processus qu'un événement important s'est produit dans son environnement d'exécution
- => un processus peut réagir à cet événement sans être obligé de le tester périodiquement

  - Mécanisme utilisé par le système pour informer les processus d'erreurs lors des appels systèmes ou d'une erreur d'instruction : overflow, adresse illégale, etc.
  - Le signal est envoyé par la routine de traitement de l'exception causée par l'erreur.
  - utilisés pour avertir un processus que l'utilisateur a frappé une touche du clavier du terminal auquel il est attaché.

Signaux

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- Tout processus peut envoyer un signal à un autre processus (moyennant certaines autorisations) par la fonction système kill().

- La réception d'un signal provoque le plus souvent la fin du processus qui le reçoit, sauf si installation préalable d'une fonction de traitement par un appel de signal().

Signaux

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- nom, numéro, signification
- SIGHUP 1 (hang up) émis à tous les processus associés à un terminal lorsque celui-ci se déconnecte.
- SIGINT 2 (interrupt) signal d'interruption émis à tous les processus associés à un terminal lorsque le caractère d'interruption (en général <CTRL-C>) est tapé.
- SIGQUIT 3 (quit) signal d'interruption émis à tous les processus associés à un terminal lorsque le caractère pour quitter une application (en général <CTRL-\>)est tapé.
- SIGILL 4 (illegal) émis en cas d'instruction illégale.
- SIGTRAP 5 (trap) émis après chaque instruction en cas de traçage de processus.

Signaux

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- SIGIOT 6 (input/output trap) émis en cas d'erreur matérielle.
- SIGKILL 9 (kill) tue un processus, quel que soit son état.
- SIGSEGV 11 (segmentation violation) émis en cas de violation de la segmentation mémoire.
- SIGSYS 12 (system) émis en cas d'erreur de paramètre dans un appel système.
- SIGPIPE 13 (pipe) émis en cas d'écriture sur un tube sans lecteur.
- SIGALRM 14 (alarm) signal associé à une horloge.
- SIGTERM 15 (termination) terminaison normale d'un processus.

Signaux

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- int kill(int pid, int sig);

- Émet le signal sig au processus pid, à condition que les proc. émetteur et destinataire soient du même propriétaire, ou que le premier soit le super-utilisateur.
- Renvoie 0 en cas de succès et -1 sinon.

Signaux

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- void (*signal(int sig, void (* fonction)(int)))(int);

- Permet d'installer fonction comme fonction de traitement lors de la réception du signal sig. La fonction de traitement peut être remplacée par une des constantes SIGIGN et SIGDFL . Le signal SIGKILL ne peut être ignoré.
- Renvoie l'adresse de la fonction de traitement précédemment utilisée en cas de succès et -1 en cas d'échec. .

Signaux

```
#include <signal.h>
void trait_sigint(){
printf("bien recu SIGINT, mais je m'en moque\n");
}
void trait_sigquit() {
printf("bien recu SIGQUIT, mais je m'en moque\n");
}
main(){
signal(SIGINT, trait_sigint);
signal(SIGQUIT, trait_sigquit);
sleep(120);
print("je meurs de ma belle mort\n");
}
```

Signaux

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

Autres appels systèmes associés aux signaux

- int pause();
- Suspend le processus jusqu'à réception d'un signal. Suivant la nature du signal et les traitements de signaux mis en place, il reprend son exécution ou traite le signal puis reprend son exécution, ou bien se termine directement.

- int alarm (int sec);
- Programme l'envoi par le système du signal SIGALRM au processus appelant dans sec secondes