Université de ***Batna 2***

Communication interprocessus : **Files de Messages (FdM)**

Département d'Informatique

Master ISI

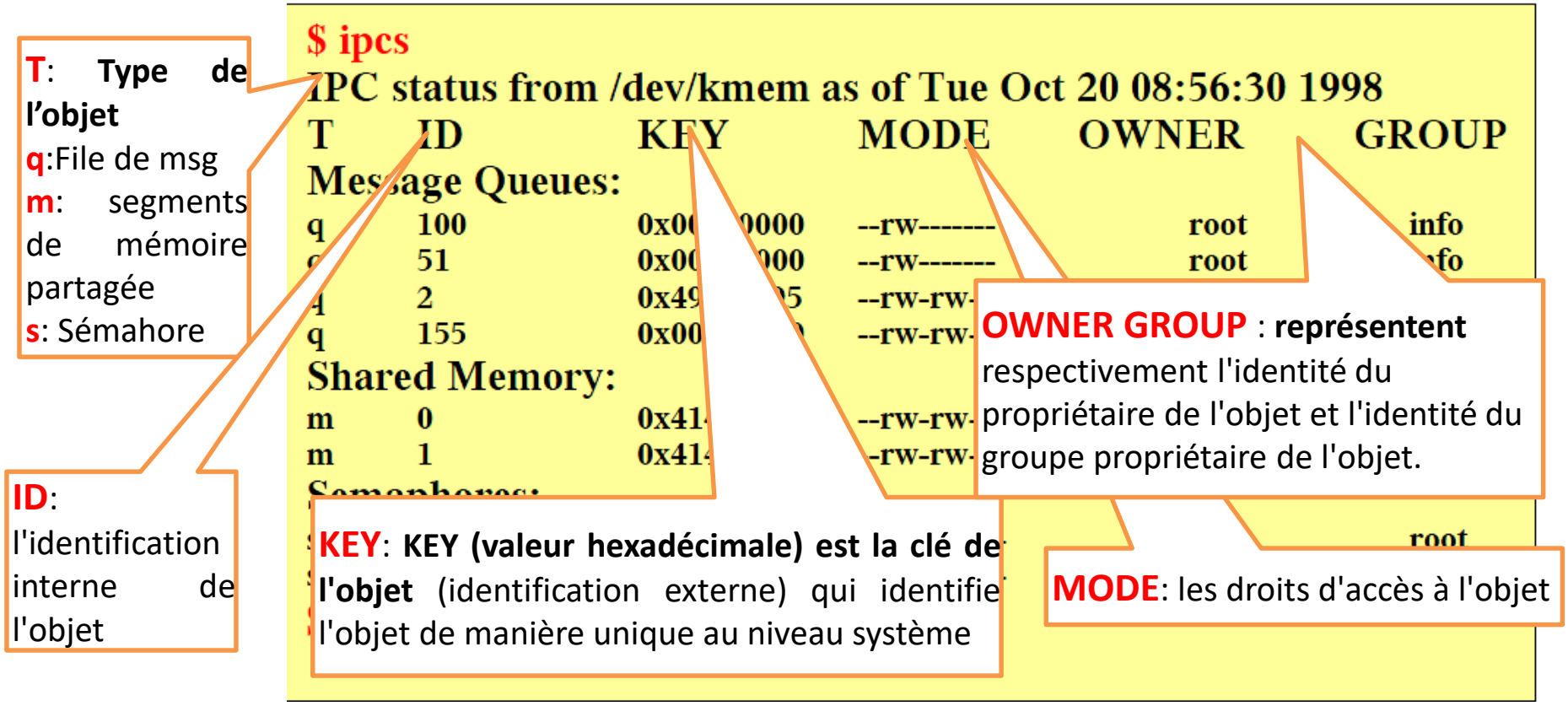
Plan

Introduction

- **Les IPC Généralités**
  - Les IPC (*Inter Process Communication) UNIX* System V présentent trois outils de communication entre des processus situés sur une même machine.
    - **Les files de messages**,
    - Les segments de mémoire partagée,
    - Les sémaphores.
- **Les commandes Shell sur les objets IPC**
  - La commande **ipcs** permet de consulter les tables systèmes,
  - La commande **ipcrm** supprime une entrée de la table.

Introduction

- **Les commandes Shell sur les objets IPC**
  - La commande **ipcs** permet de consulter les tables systèmes,



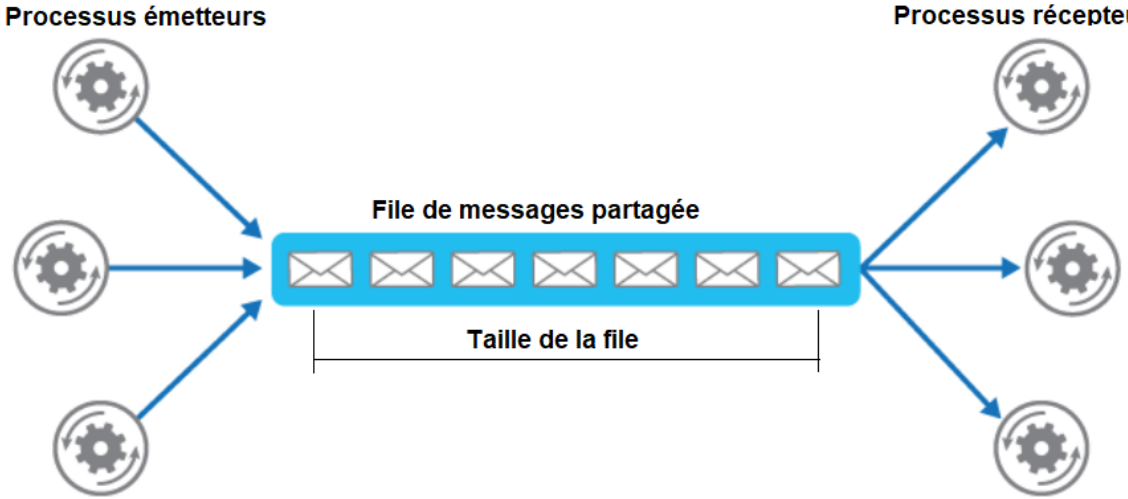
Introduction

- **Les IPC Généralités**
  - Pour partager un objet IPC, les processus partagent la clé externe qui lui est associée et utilisent les méthodes propres à chaque type d'objet IPC.
  - Chaque objet **IPC** possède :
    - Un **identificateur** : (ID) équivalent pour les fichiers aux descripteurs dans la table des fichiers ouverts de chaque processus.
    - Une **clé** :équivalente aux références (noms symboliques ou chemin + nom) pour les fichiers.
  - Pour qu'un processus utilise un objet IPC, il doit connaître son **ID**. Pour cela il utilise la ***clé externe*** *associée* à cet objet.

- **Dans ce qui suit, nous nous intéresserons uniquement aux Filles de Messages (FdM)**

Files de Messages : Principe

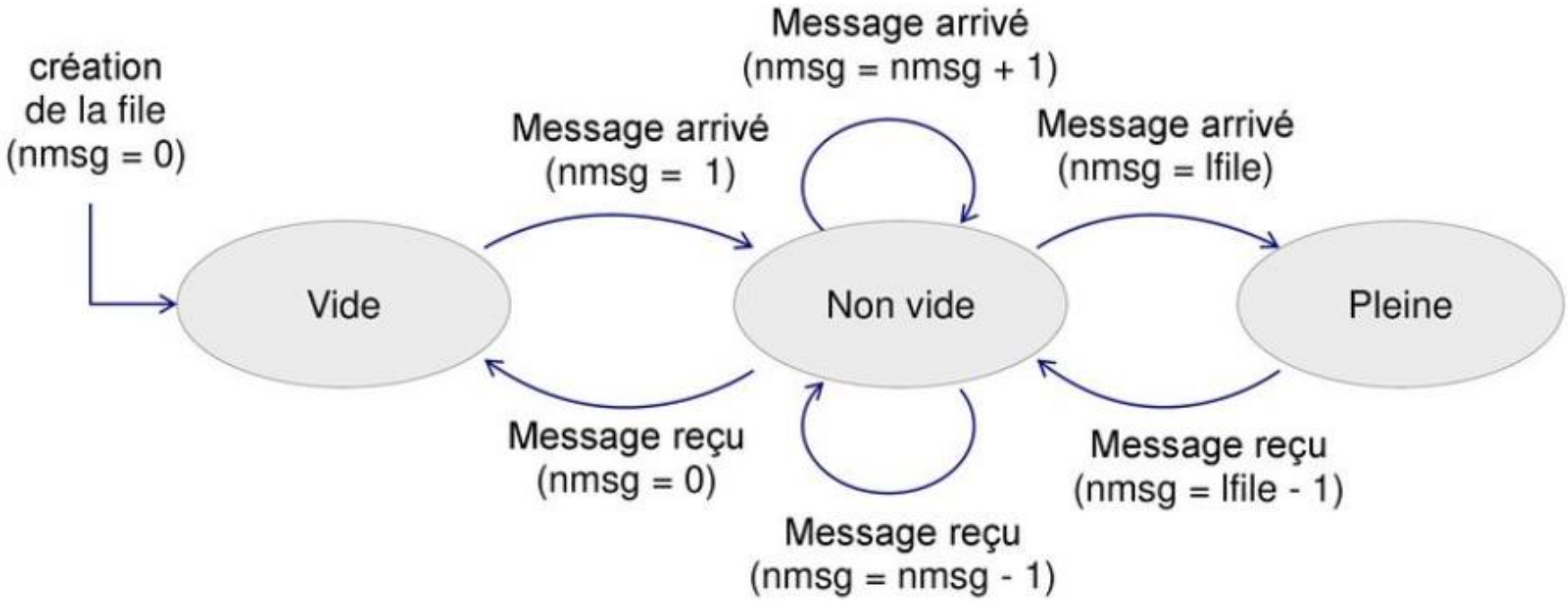
- Il s’agit d’une implémentation du principe de la boîte aux lettres.



- Un processus a la possibilité d'y déposer des messages ou d'en extraire.

Files de Messages : Etats

- **États des FdM**



Files de Messages

- **Caractéristiques d'un message**
  - Un message est un couple formé d'un nombre entier (type) et d'une suite d'octets de longueur arbitraire constituant le message proprement dit.

  - Structure générique d'un message

```
struct msgbuf {
        long  mtype ;     /* type du message */
        char mtext[1] ;  /*texte du message */
}
```

  - Grâce au type, un processus récepteur pourra distinguer parmi les messages d'une file ceux qu'il souhaite extraire.
  - Un processus récepteur peut choisir de se mettre en attente soit :
    - sur le **premier message disponible**,
    - sur le **premier message d'un type donné**.

Files de Messages : Principe

- **Caractéristiques d'un message**
  - Un processus peut émettre des messages même si aucun processus n'est prêt à les recevoir.
  - Les messages déposés sont conservés après la mort du processus émetteur, jusqu'à leur consommation ou la destruction de la file.
  - Les messages ne contiennent pas à priori le numéro du processus émetteur ni celui du destinataire.
    - c'est aux programmeur de décider de l'interprétation de chaque message
  - Le message forme un tout:
    - On le dépose ou on l'extrait en une seule opération.

Files de Messages : Caractéristiques

- **Clef & Descripteur**
  - une clef (définie par le créateur), qui pourra être utilisée par les autres processus pour accéder à la file.
  - un descripteur (identificateur), retourné à la création, et utilisé pour les manipulations de la file au sein du processus.
  - lors de la création, une structure msqid_ds est créée, qui correspond à une entrée dans la table des files de messages gérées par le système. Cette structure identifie la file.

Files de Messages : Opérations

- **Opérations sur les FdM**
  - Pour pouvoir utiliser une FdM, on doit disposer des primitives suivantes :
    - **création et ouverture** d'un file de message : ***msgget()***
    - **gestion ou contrôle** de la file de messages : ***msgctl()***
    - **envoi ou réception** des messages : ***msgsnd()*** *et* ***msgrcv()***

Files de Messages: Opérations

- **Création d'une FdM**

```
#include <sys/msg.h>
int  msgget (key_t  cle, int  options);
```

  - retourne **l'identificateur** de la file (ou **-1** en cas d'erreur).
  - Le paramètre options doit être construite avec une conjonction du mode d'accès et des constantes IPC_CREAT et IPC_EXCL.
  - Si cle == IPC_PRIVATE
    - Une nouvelle file de messages privée est créée.
    - Cela est suffisant lorsque les processus qui sont reliés par un lien de parenté.
  - Le paramètre options peut valoir :
    - IPC_CREAT : créer la file si elle n'est pas créée.
    - IPC_EXCL positionnée en même temps que IPC_CREAT pour sortir en erreur si l'on essaie de créer une file existante.
    - MSG_R : permission en lecture, MSG_W : permission en écriture

Files de Messages : Opérations

- **Envoie d'un message**

#include <sys/types.h> #include <sys/ipc.h> #include <sys/msg.h>

int **retour** = msgsnd (int msg_id, void *pt_msg, int size, int option);

  - int msg_id : identifiant de la file de messages. Il est retourné par ***msgget****().*
  - *pt_msg : pointeur universel *vers une variable contenant le message à envoyer.* Il est de type ***struct msgbuf*** *créée et remplie par le programmeur.*
  - **size** : taille du corps du message.
  - option : option d'envoi du message. Celle-ci peut être :
    - IPC_NOWAIT : l'appel à la fonction n'est pas bloquant. Si le message ne peut pas être envoyé (pas de place, etc.), la fonction retourne (*-1)*
    - 0 : l'appel est bloquant (standard). Tant que le message ne part pas, la fonction reste en attente.
  - **retour** : taille en octets du message déposé dans la FdM.

Files de Messages : Opération

- **Réception d'un message**

int **msgrcv** (int msqid, struct msgbuf *msgp, int msgsz, long msgtyp, int msgflg);

  - msqid : identificateur de la FdM
  - msgp : zone de récupération du message à lire dans la FdM
  - msgsz : taille maximale de la zone mémoire pointée par msgp
  - msgtyp : type du message à lire ;
    - 0 signifie que l'on prend le 1er message de la FdM quelque soit son type (le plus vieux dans FIFO).
    - Si Msgtyp =n >0 : Lire le msg le plus ancien contenant dans le champ type la valeur n.
    - Si Msgtyp =n <0 : Lire le msg le plus ancien contenant dans le champ type la plus petite valeur.
  - msgflg : option de fonctionnement du processus appelant (IPC_NOWAIT évite l'attente passive).

Files de Messages : Opérations

- **Contrôle**

int **msgctl** (int msqid, int cmd, struct ms_qid_dsn *buf);

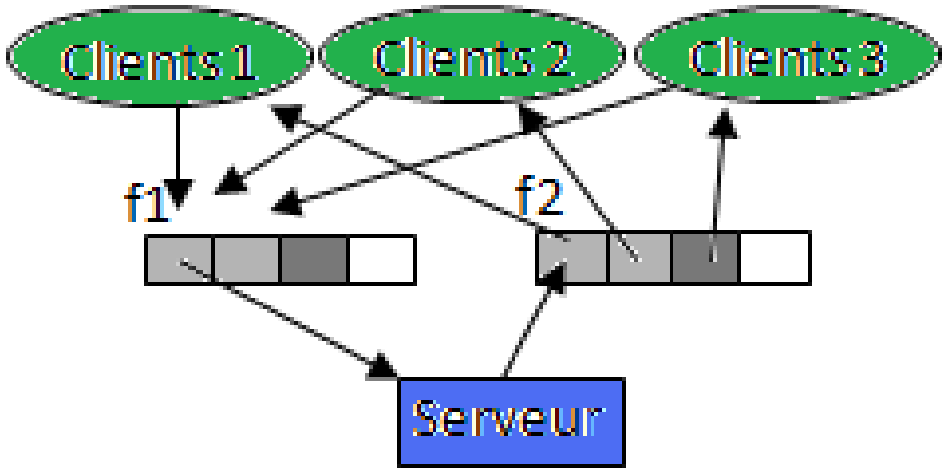
  - msqid : identificateur de la FdM visée
  - cmd : type d'opération à effectuer sur la FdM, par exemple IPC_RMID pour détruire la FdM
  - buf : pour récupérer la table associée à la FdM

Files de Messages : Cas pratique

- **Exemple**

  Soit un processus serveur (cyclique) dédié aux calculs numériques (+,-,*) ainsi qu'un nombre quelconque de processus *clients* qui sollicitent le *serveur* pour effectuer des calculs de type : A op B. Le *serveur* exécute les calculs dictés puis renvoie les résultats aux différents *clients* demandeurs.

  Deux FdM (f1 et f2) sont utilisées pour garantir la communication entre le *serveur* et ses *clients*



Files de Messages : Cas pratique

- **Exemple**
  - Règles de fonctionnement
    - Une requête est envoyée par un ***client*** au ***serveur*** grâce un message de type **m1** via la FdM **f1**.
    - Le message **m1** est composé du **PID** du ***client***, les opérandes A et B et l'opération à effectuer.
    - L'opération peut être une addition (+), une soustraction (-) ou une multiplication (*).
    - Le ***serveur*** lit les messages en FIFO
    - Pour chaque message
      - le ***serveur*** exécute l'opération spécifiée dans le message sur les deux opérandes A et B mentionnés aussi dans le message et retourne le résultat à travers un message de type **m2** déposé dans la deuxième FdM **f2**.
    - Le message **m2** doit contenir aussi le **PID** du ***client***.

Files de Messages : Cas pratique

- **Exemple**
  - ***Déclaration des messages m1 et m2***

```
typedef struct {
        long pid; /* pid du processus émetteur*/
        int a ;
        int b ;
        char op ;
 } m1 ;

typedef struct {
        long pid; /* pid du processus concerné par le msg*/
        int resultat ;
} m2 ;
```

Files de Messages : Cas pratique

- **Exemple**
  - ***Création des FdM f1 et f2***

```
/* Génération des clés*/
    key_t cle1= ftok("fichier1",1) ;
    key_t cle2= ftok("fichier2",2) ;

/*Création des FdM */
    int fd1= msgget (cle1, IPC_CREAT|IPC_EXCL)
    int fd2= msgget (cle2, IPC_CREAT|IPC_EXCL)
```

Files de Messages : Cas pratique

- **Exemple**
  - ***Corps du serveur***

```
while (1) {
        /* Lire un msg */
      int retourne = msgrcv (fd1, &mess_rcv, sizeof(m1), 0,0) ;
/* calculer l'opération sollicitée */
      switch (mess_rcv.op){
            case'*' : mess_snd.resultat= mess_rcv.a * mess_rcv.b ;
                      break ;
           case '+' : mess_snd.resultat= mess_rcv.a + mess_rcv.b ;
                      break ;
           case '-' : mess_snd.resultat= mess_rcv.a - mess_rcv.b ;
                      break ;
           default : printf("erreur de l'opération \n") ;
      }
      mess_snd.pid = mess_rcv.pid ;
      /* envoyer le msg résultat*/
      retourne = msgsnd(fd2,&mess_snd, sizeof(m2),0) ;
}
```

Fin

Merci pour votre attention