Les Threads

LaBRI, UMR CNRS 1308
Université Bordeaux 1

Serge Chaumette
Serge.Chaumette@labri.fr

Objectifs de ce cours

- Objectifs
  - Maîtriser les principes des Threads et de l'exclusion mutuelle
  - Connaître :
    - Les Threads Posix
    - Les Threads Java
    - Les Threads NT
- Évaluation
  - Exercices
  - Manipulations

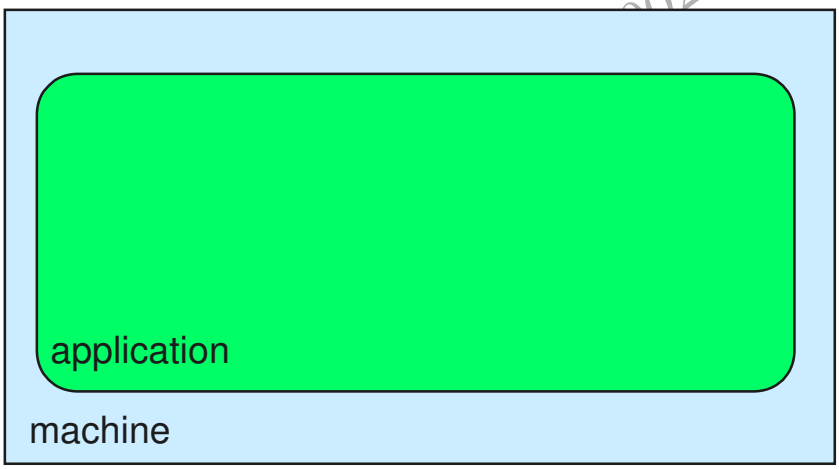
© Serge Chaumette, 2001, 2002

Contenu

- Évolution de la multiprogrammation
- Les processus
- Les *threads*
- Les sémaphores

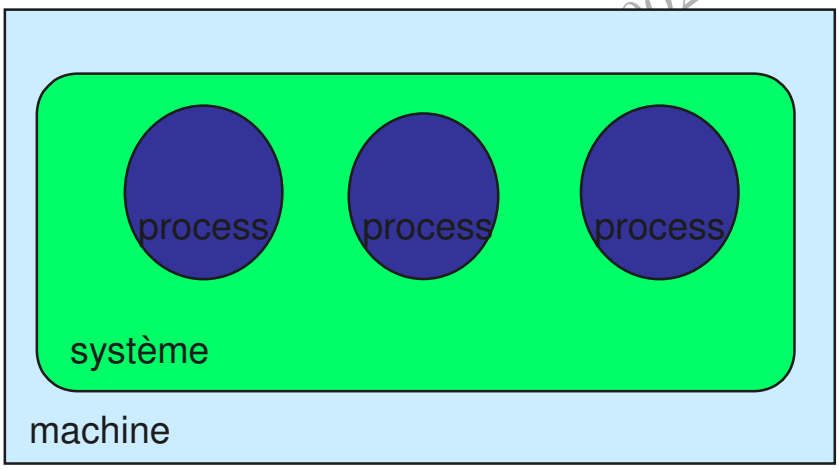
© Serge Chaumette, 2001, 2002

La monoprogrammation



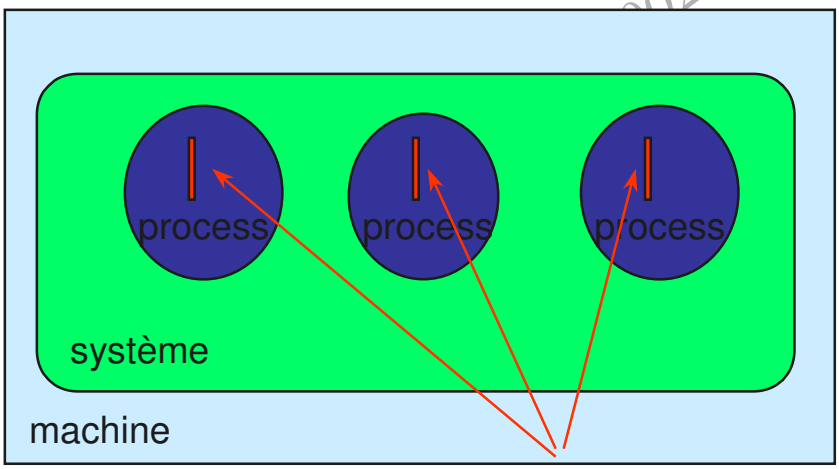
[Evolution de la multiprogrammation]

La multiprogrammation ou multiprocessus



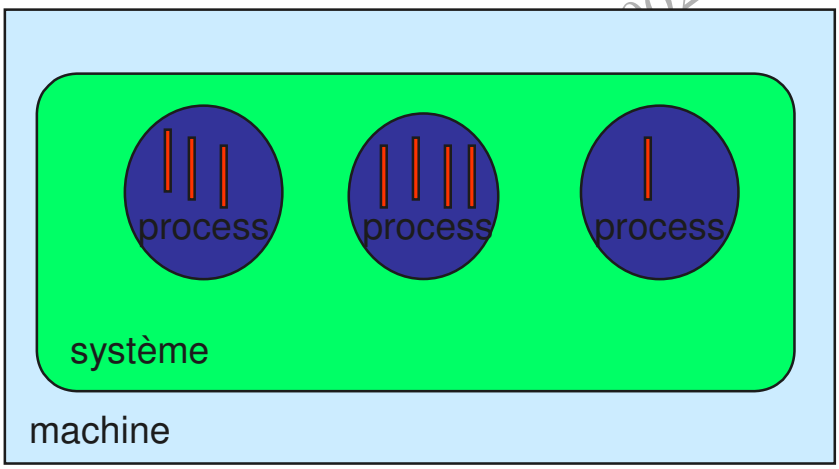
[Evolution de la multiprogrammation]

La multiprogrammation ou multiprocessus



1 *thread* / processus

Le multi*thread*

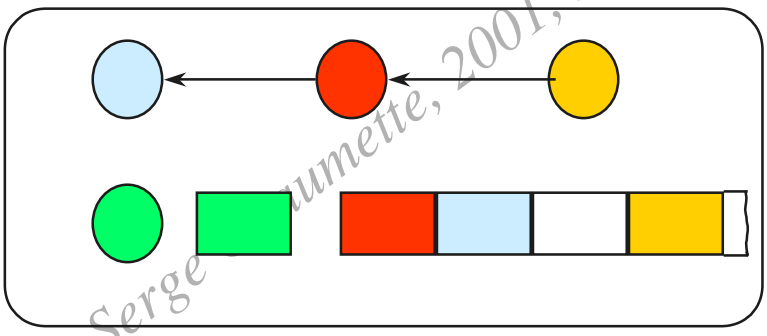


Les processus

- Processus et système
- Changement de processus
- API POSIX

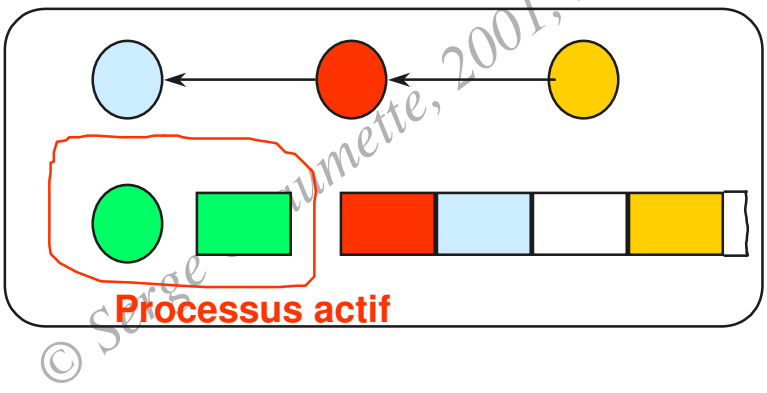
© Serge Chaumette, 2001, 2002

Processus et système



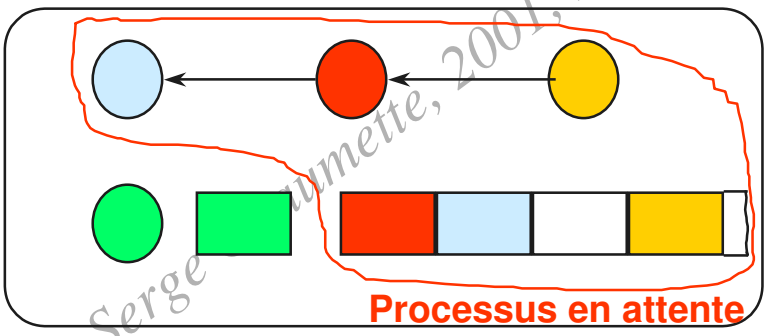
© Serge Chaumette, 2001, 2002

Processus et système



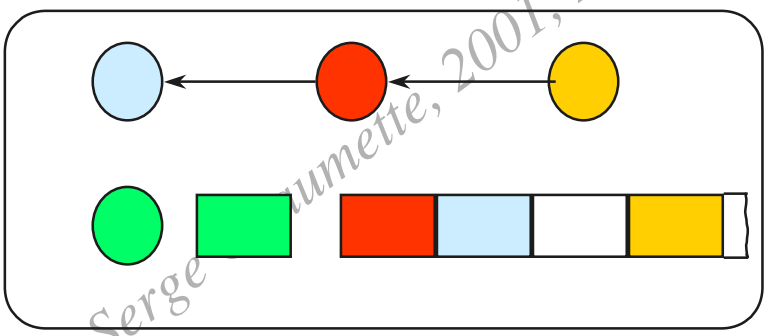
© Serge Chaumette, 2001, 2002

Processus et système



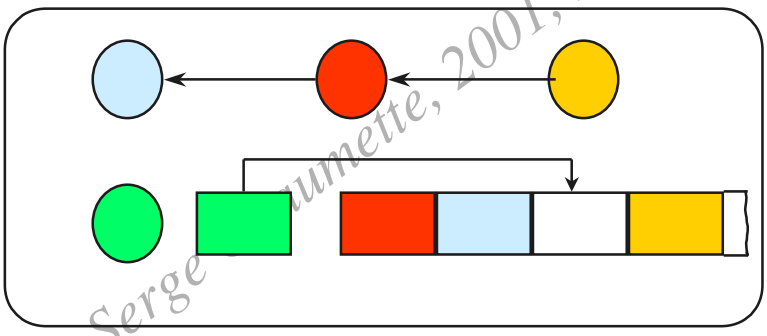
© Serge Chaumette, 2001, 2002

Changement de processus



© Serge Chaumette, 2001, 2002

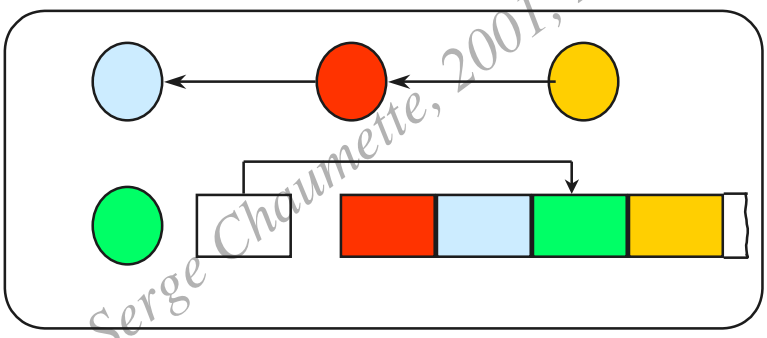
Changement de processus



© Serge Chaumette, 2001, 2002

sauvegarde des structures système

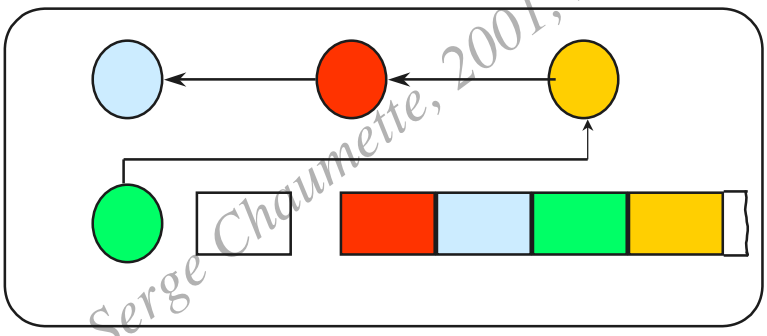
Changement de processus



© Serge Chaumette, 2001, 2002

sauvegarde des structures système

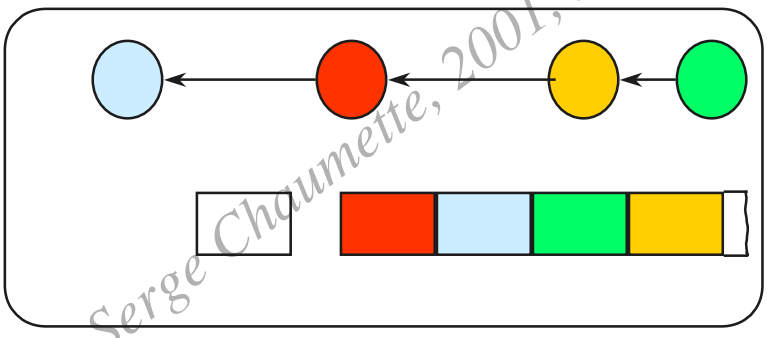
Changement de processus



© Serge Chaumette, 2001, 2002

mise en attente du processus

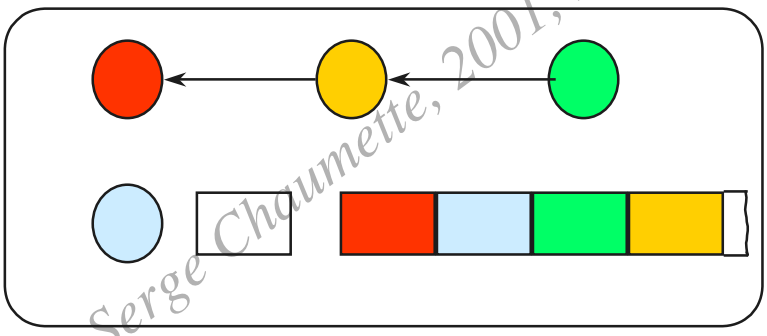
Changement de processus



© Serge Chaumette, 2001, 2002

mise en attente du processus

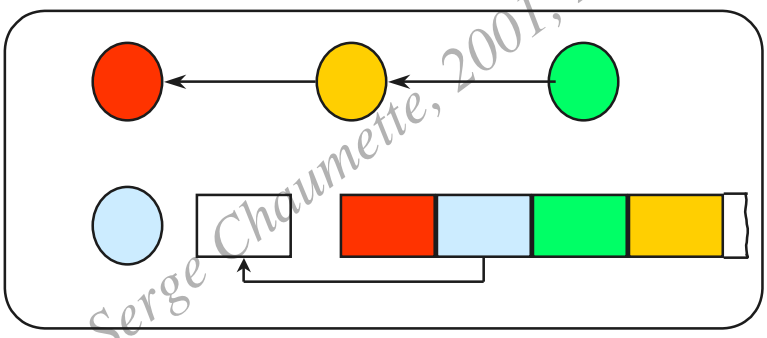
Changement de processus



© Serge Chaumette, 2001, 2002

choix d'un autre processus

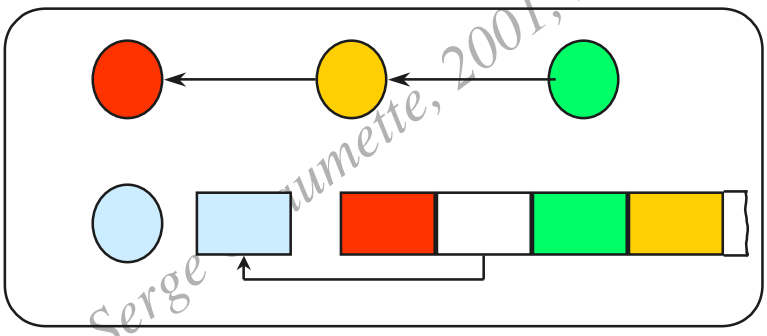
Changement de processus



© Serge Chaumette, 2001, 2002

restauration des structures système

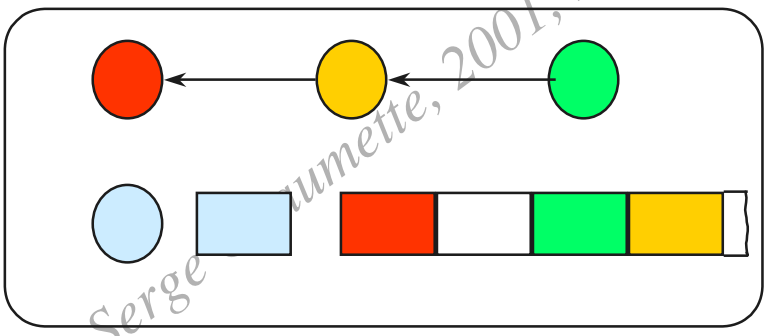
Changement de processus



© Serge Chaumette, 2001, 2002

restauration des structures système

Changement de processus



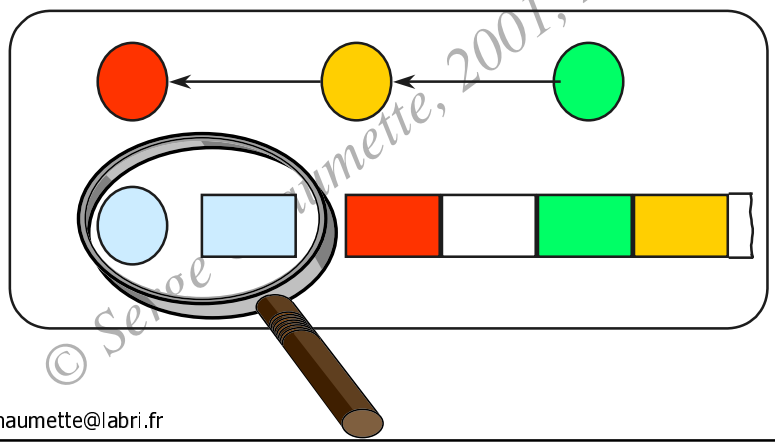
© Serge Chaumette, 2001, 2002

Les *threads*

- *Threads* et processus
- Changement de *thread*
- Exemple
- API POSIX

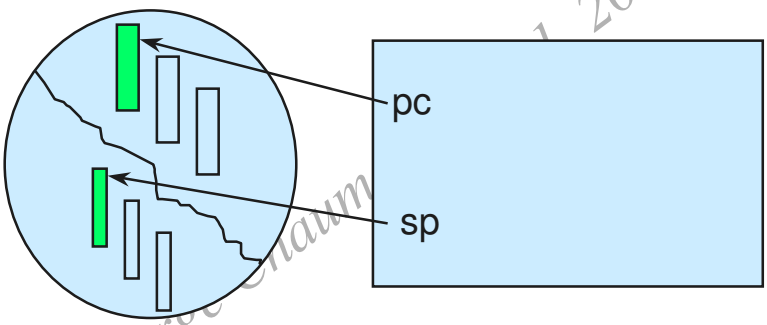
© Serge Chaumette, 2001, 2002

Threads et processus



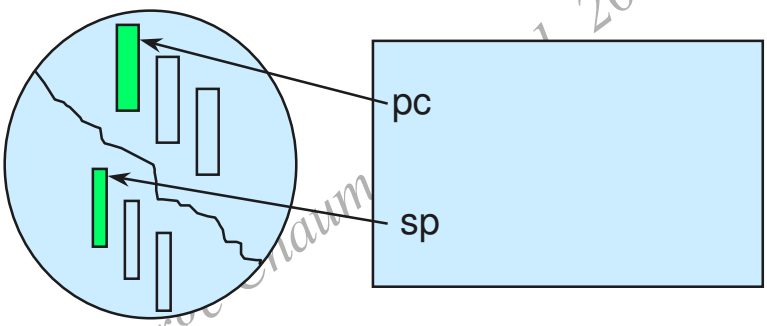
© Serge Chaumette, 2001, 2002

Threads et processus



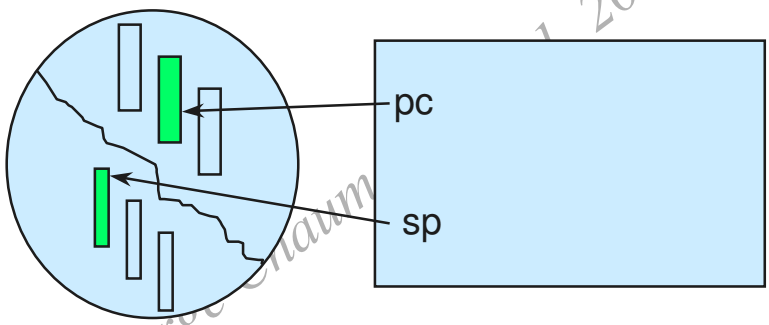
© Serge Chaumette 2002

Changement de *thread*



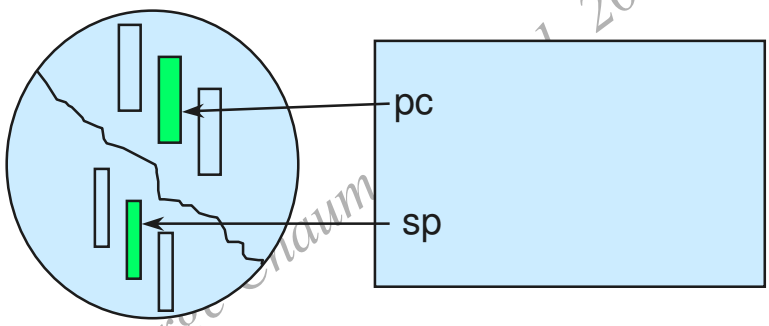
© Serge Chaumette 2002

Changement de *thread*



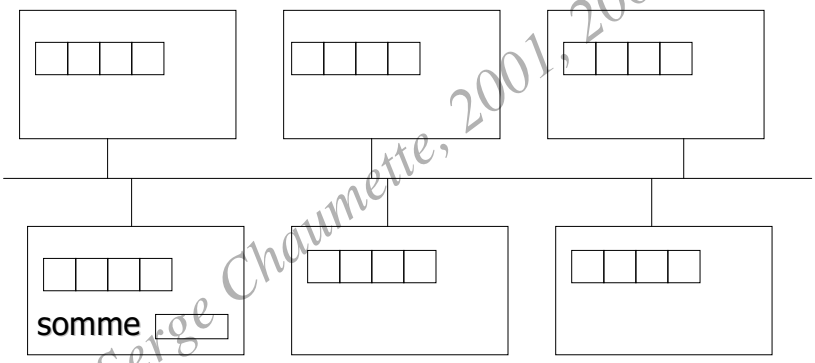
© Serge Chaumette 2002

Changement de *thread*



© Serge Chaumette 2002

Exemple



© Serge Chaumette, 2001, 2002

Exemple

```
a s. ⟶ pour i de 1 a longueur(vecteur)
b s. ⟶   si (proprietaire(vecteur[i])==moi)
             alors
c s. ⟶         somme=somme+vecteur[i];
             sinon
d s. ⟶         recevoir(proprietaire(vecteur[i]), valeur)
c s. ⟶         somme=somme+valeur;
         fin si
       fin pour
```

© Serge Chaumette, 2001, 2002

Exemple

```
a s. → pour i de 1 a longueur(vecteur)
b s. →   si (proprietaire(vecteur[i])==moi)
           alors
c s. →       somme=somme+vecteur[i];
           sinon
d s. →       recevoir(proprietaire(vecteur[i]), valeur)
c s. →       somme=somme+valeur;
         fin si
       fin pour
```

durée locale DLocale= *a+b+c*

Serge.Chaumette@labri.fr

© Serge Chaumette, 2001, 2002

[Les *threads*]

Exemple

```
a s. → pour i de 1 a longueur(vecteur)
b s. →   si (proprietaire(vecteur[i])==moi)
           alors
c s. →       somme=somme+vecteur[i];
           sinon
d s. →       recevoir(proprietaire(vecteur[i]), valeur)
c s. →       somme=somme+valeur;
         fin si
       fin pour
```

durée distante DDistante= *a+b+d+c*

Serge.Chaumette@labri.fr

© Serge Chaumette, 2001, 2002

Exemple

durée locale DLocale= *a*+*b*+*c*

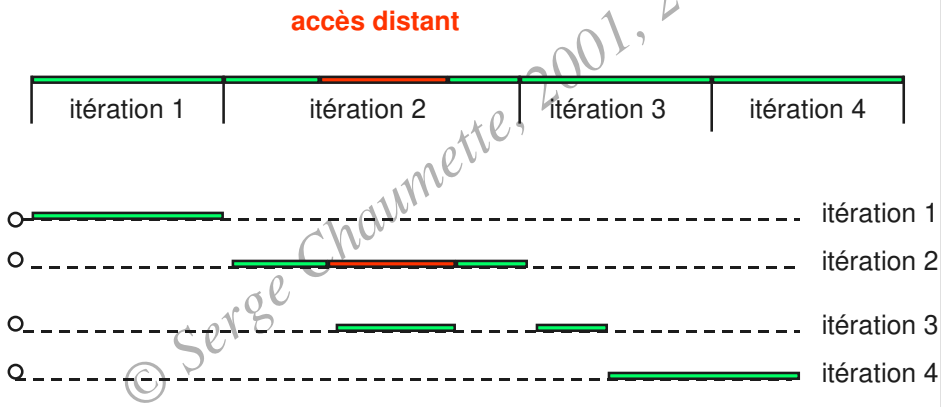
durée distante DDistante=*a*+*b*+*d*+*c*

pendant la durée *d* l'application est bloquée

durée totale DTotale = #locales*DL + #distantes*DD

© Serge Chaumette, 2001, 2002

Exemple



© Serge Chaumette, 2001, 2002

Exemple

durée locale DLocale= *a*+*b*+*c*

durée distante DDistante=*a*+*b*+*d*+*c*

pendant la durée *d* l'application n'est pas bloquée

durée totale DT = #*DL

© Serge Chaumette, 2001, 2002

Ecriture du code des *Threads*

f1(<*parametre*>){
<*instructions*>
}

f2(<*parametre*>){
<*instructions*>
}

f3(<*parametre*>){
<*instructions*>
}

© Serge Chaumette, 2001, 2002

API POSIX

```
#include <pthread.h>

int pthread_attr_init (pthread_attr_t *attr);

int pthread_create(
        pthread_t *new_thread_ID,
        const pthread_attr_t *attr,
        void * (*start_func)(void *), void *arg
        );
void pthread_exit (void *status);
```

© Serge Chaumette 2001, 2002

un seul paramètre → utilisation d'une structure

API POSIX

```
#include <pthread.h>

void main(void){

  pthread_attr_t attr;
  pthread_t new_thread_ID;
  int value=10;

  pthread_attr_init (&attr);
  pthread_create(&new_thread_ID ,
                 &attr,
                 f,  (void *) &value);
}
```

```
#include <pthread.h>

void f(void *arg){

  printf("%d\n", *(int *) arg);
  pthread_exit (NULL);
}
```

Les sémaphores

- Intérêt
- Présentation intuitive
- Principe
- Exemple
- API POSIX

© Serge Chaumette, 2001, 2002

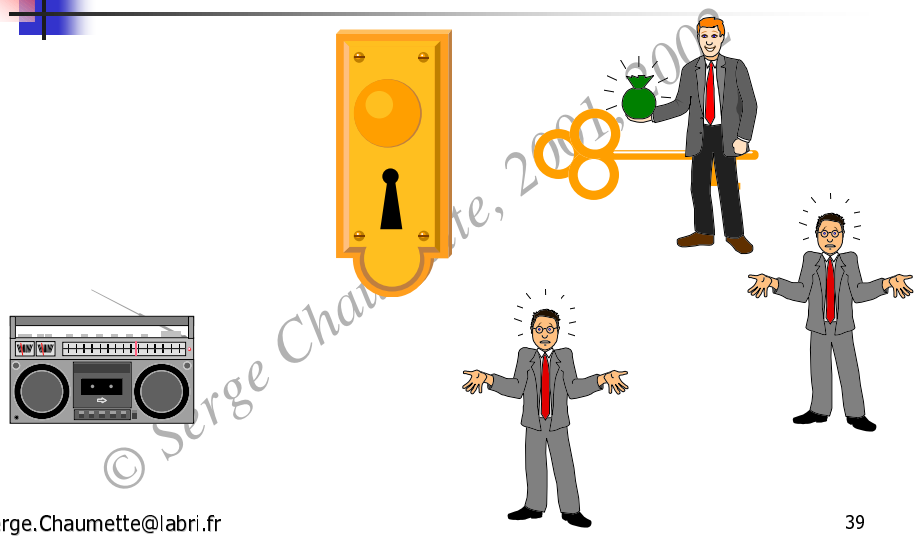
Intérêt

accès exclusif à une ressource

- variables partagées
- fichiers
- réseau de communication
- autres périphériques
- ...

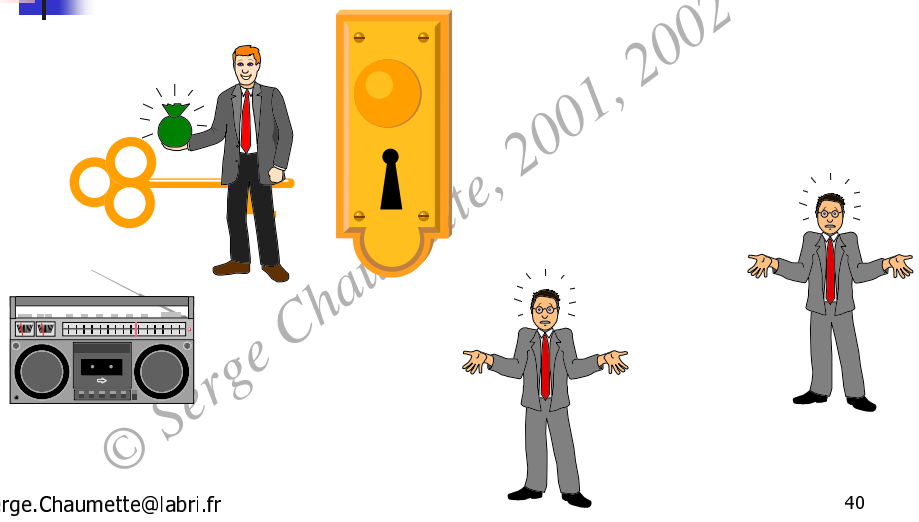
© Serge Chaumette, 2001, 2002

Présentation intuitive



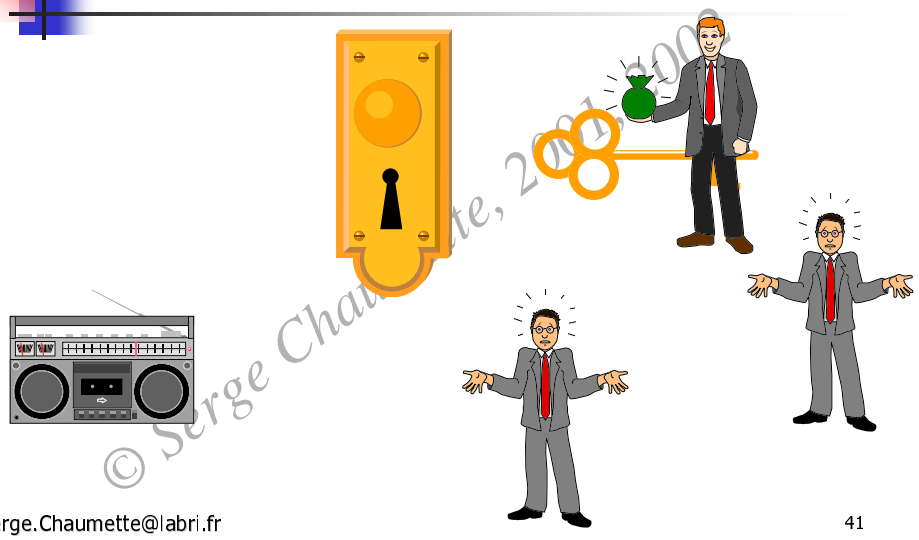
© Serge Chaumette, 2001, 2002

Présentation intuitive



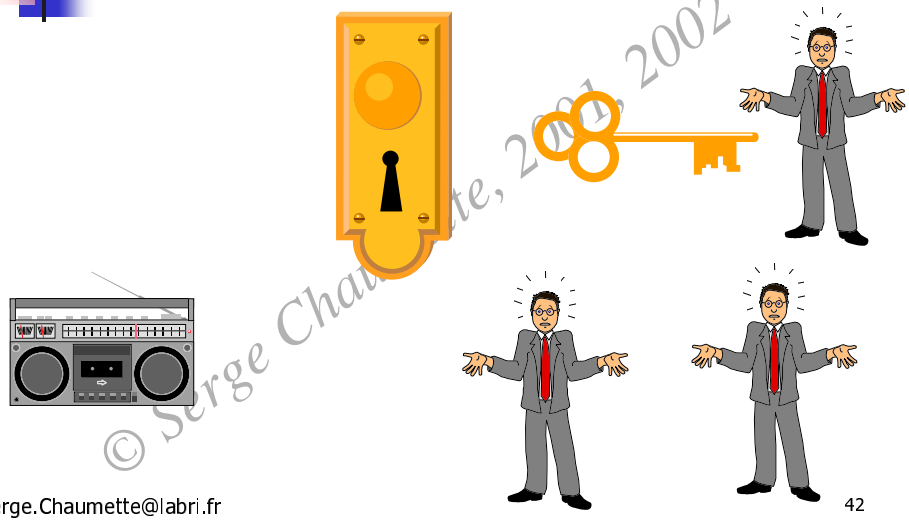
© Serge Chaumette, 2001, 2002

Présentation intuitive



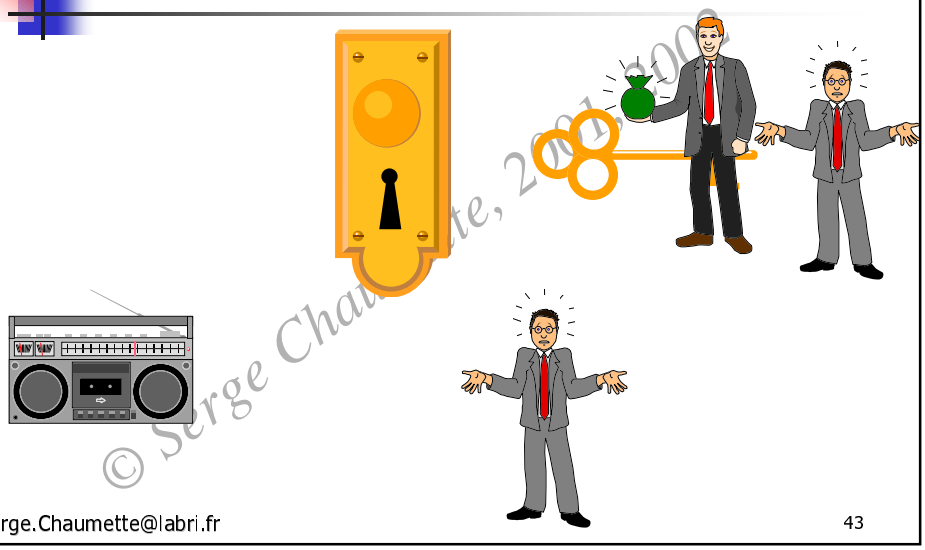
© Serge Chaumette, 2001, 2002

Présentation intuitive



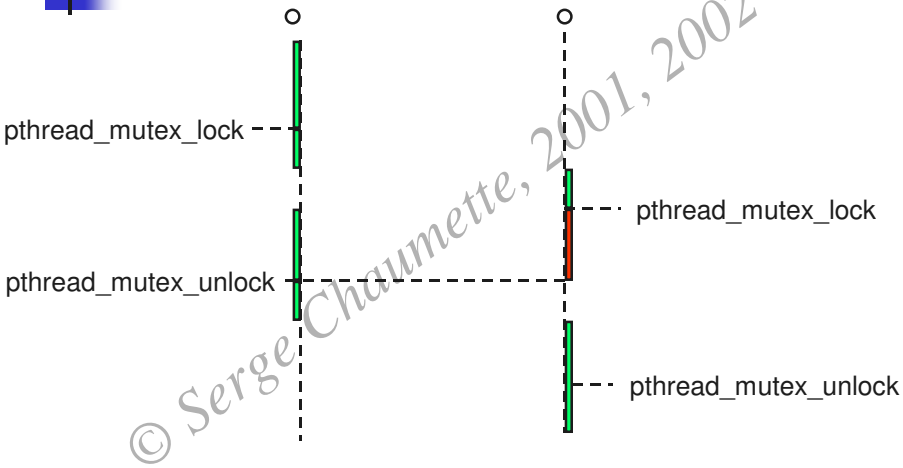
© Serge Chaumette, 2001, 2002

Présentation intuitive



© Serge Chaumette, 2001, 2002

Principe



© Serge Chaumette, 2001, 2002

Exemple

```
pour i de 1 a longueur(vecteur)
    si (proprietaite(vecteur[i])==moi)
        alors
            somme=somme+vecteur[i];
        sinon
            recevoir(proprietaire(vecteur[i]), valeur)
            somme=somme+valeur;
    fin si
fin pour
```

© Serge Chaumette, 2001, 2002

Exemple

```
somme=somme+vecteur[i];
```

```
load registre1, vecteur[i];
load registre 2, somme;
add;
store somme, registre 3
```

```
load registre1, vecteur[j];
load registre 2, somme;
add;
store somme, registre 3
```

© Serge Chaumette, 2001, 2002

Exemple

somme=somme+vecteur[i];

load registre1, vecteur[i]; 1	load registre1, vecteur[j];3
load registre 2, somme; 2	load registre 2, somme; 4
add; 7	add; 5
store somme, registre 3 8	store somme, registre 3 6

on perd vecteur[j]

© Serge Chaumette, 2001, 2002

Exemple

somme=somme+vecteur[i];

load registre1, vecteur[i];	load registre1, vecteur[j];
lock	lock
load registre 2, somme;	load registre 2, somme;
add;	add;
store somme, registre 3	store somme, registre 3
unlock	unlock

© Serge Chaumette, 2001, 2002

API POSIX

```
#include <pthread.h>

int pthread_mutex_init(
        pthread_mutex_t *mp,
        const pthread_mutexattr_t *attr);

int pthread_mutex_lock(pthread_mutex_t *mp);
int pthread_mutex_trylock(pthread_mutex_t *mp);
int pthread_mutex_unlock(pthread_mutex_t *mp);
int pthread_mutex_destroy(pthread_mutex_t *mp);
```

© Serge Chaumette, 2001, 2002

Initialisation

```
pthread_mutex_t mutex;
pthread_mutexattr_t attr;

pthread_mutexattr_init(&attr);
pthread_mutex_init(&mutexp, &attr);
```

© Serge Chaumette, 2001, 2002

[Exercice]

Recouvrement calcul/communication

15 minutes

- Proposer une architecture à base de *threads* qui permettrait à un navigateur Web de pré-fetcher les documents référencés par les URLs du document en cours de visualisation, tout en conservant un fort niveau d'interactivité avec l'utilisateur.

© Serge Chaumette, 2001-2002

[Exercice]

Ordonnanceur de *Threads*

20 minutes

- Concevoir et implémenter un système permettant d'ordonnancer les *threads* d'une application. On envisagera deux cas:
  - on dispose de fonctions stop/resume et l'ordonnanceur est préemptif
  - on ne dispose pas de ces fonctions et l'ordonnanceur n'est pas préemptif

© Serge Chaumette, 2001-2002