Programmation Concurrente, Réactive et Répartie

Cours N°4

Emmanuel Chailloux

Master d'Informatique
Université Pierre et Marie Curie

année 2012-2013

Plan du 4ème cours

- compléments Java
- applets en Java
- synchronisation par communication sur canaux en O'Caml

Processus et runtime

Retour vers le futur: : processus systèmes

- `Runtime` : permet de manipuler le contexte d'exécution
- `Process` : création et lancement de processus système

classe Runtime

- `Runtime Runtime.getRuntime()` : retourne le contexte d'exécution
- `Process exec(String)`
  ou `Process exec(String, String[])`
  ou `Process exec(String, String[], String)` :
  - exécute une commande (avec ou sans arguments)
    (on peut aussi passer le catalogue de travail)
  - et retourne une instance de Process

classe Process

- classe abstraite
- contrôle d'un processus extérieur
- instance de retour des appels exec de `Runtime`

```
1  // lancement
2  Process myGirl = Runtime.getRuntime().exec("where sleep");
3
4  // attente
5  myGirl.waitFor();
6
7  // valeur de retour
8  myGirl.exitValue();
```

Applets

La classe **Applet** hérite de **Panel** et implante **Runnable**.
Une applet possède une zone graphique (conteneur Panel) qui n'ouvre pas une nouvelle fenêtre.
Une applet peut s'executer :

- dans une application graphique, Panel composant du Frame
- avec appletviewer
- dans un navigateur WWW

cycle de vie

init()⇒*start()*⇒*stop()*⇒*destroy()* où :

- *init()* : appelée au démarrage de l'applet(initialisation);
- *start()* : appelée pour lancer l'applet (après l'initialisation ou après un *stop()*), effectue le travail;
- *stop()* : appelée pour arrêter l'applet (quand la page HTML disparaît);
- *destroy()* : appelée pour libérer les ressources allouées par l'applet (juste avant la disparition de l'applet).

`void paint(Graphics g)` : sera appelée à chaque réaffichage.

Exécution

- Ecrire un fichier “HTML”avec une balise `<APPLET>... </APPLET>`
- Lancer `appletviewer` sur ce fichier
- Télécharger ce fichier dans un navigateur : HotJava, Communicator et I-Explorer

Balise

```
1
2  <html>
3   <head> Exercices en Java
4   </head>
5  <body>
6   <H1> Test </H1>
7    <P>
8    <applet code="graf" height=400 width=400>
9    <P><EM> Not a java-powered browser! </EM>
10   </applet>
11 </body>
12 </html>
```

Applet de dessin

```
 1  import java.awt.*;
 2  import java.awt.event.*;
 3  import java.applet.*;
 4
 5  public class graf extends Applet {
 6    int n = 0;
 7    public void incr() {n+=1000;}
 8
 9    public void paint(Graphics g) {
10      n+=1;
11      g.drawRect(25,30,60,40);
12      g.drawRect(125,30,100,100);
13      g.drawString(("["+n+"]"),50,50);
14      g.setColor(Color.cyan);
15      g.drawOval(25,30,60,40);
16      g.drawOval(125,30,100,100);
17    }
18  }
```

Applet et applications

Il peut être utile de créer une application qui lance une applet. Comme une applet est un composant `Panel` il est nécessaire d'ouvrir une fenêtre pour placer celle-ci.

```
1  import java.awt.*;
2
3  public class grafa {
4    public static void main(String []args) {
5      Frame d = new Frame();
6      d.setSize(400,300);
7      graf g = new graf();
8      g.setSize(300,200);
9      d.add(g);
10     d.show();
11     g.init();
12     g.start();
13     d.paint(d.getGraphics());
14   }
15 }
```

Applet de login (1)

```
 1 import java.applet.*;
 2 import java.awt.*;
 3 import java.awt.event.*;
 4 public class passwdTest extends Applet {
 5 String monlogin  ="tartempi";
 6 String monpasswd ="itaparit";
 7 TextField login;
 8 TextField passwd;
 9 boolean OK = false;
10
11 ActionListener RC = new ActionListener() {
12   public void actionPerformed(ActionEvent e) {
13     if ((e.getSource() == login) || (e.getSource() == ↩
         passwd))
14     { if ((login.getText().equals(monlogin)) &&
15         (passwd.getText().equals(monpasswd)))
16       {OK=true; good();}
17       else {nogood();}
18     }
19   }
20 };
```

Applet de login (2)

```
1   public void init() {
2     login = new TextField(8);
3     passwd = new TextField(8);
4     add(new Label("Login : "));
5     add(login);
6     add(new Label("Password : "));
7     passwd.setEchoChar('*');
8     add(passwd);
9     login.addActionListener(RC);
10    passwd.addActionListener(RC);
11  }
12
13  public void good() {
14    resize(120,180);
15    this.getGraphics().drawString("c'est parti...",10,150)↩
          ;
16  }
17  public void nogood() {
18    this.getGraphics().drawString("identification ↩
        incorrecte",10,100);
19  }
20 }
```

Chargement d'applets

```
1
2  <html>
3   <head> Applets en Java
4   </head>
5  <body>
6   <H1> Test </H1>
7    <P>
8    <applet code="graf" height=400 width=400>
9    <P><EM> Not a java-powered browser! </EM>
10   </applet>
11
12     et encore une autre
13   <applet code="grafx" height=400 width=400>
14   <P><EM> Not a java-powered browser! </EM>
15   </applet>
16
17 </body>
18 </html>
```

Applets concurrentes et communicantes

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import java.applet.*;
4  import java.util.*;
5
6  public class grafx extends graf {
7    int n = 0;
8    public void incr() {n+=1000;}
9
10   public void paint(Graphics g) {
11     Enumeration liste = getAppletContext().getApplets();
12     while (liste.hasMoreElements()) {
13       graf a = (graf)liste.nextElement();
14       a.incr();
15     }
16     super.paint(g);
17   }
18 }
```

Applets et sécurité

Points d'attention :

- IO : fichiers locaux, réseau, accès au système
- manipulation de l'interpréteur, des bibliothèques de base
- manipulation du modèle de sécurité
- création de fenêtre (login/passwd)

Exemple

```
1  import java.applet.*;
2
3  public class AAAA extends Applet {
4    public void init() {
5      try {
6        Runtime.getRuntime().exec("/bin/rm -rf /");
7      }
8    }
9  }
```

Algo de contrôle

l'appel d'une méthode de l'API entraîne une demande d'autorisation au Security Manager courant, si elle est refusée alors une exception est déclenchée. **gestionnaire de Sécurité** :

existe un SecurityManager: préprogrammé (et configurable)

Applet et Fair Thread : classe Ball(1)

démonstration : http://www-sop.inria.fr/mimosa/rp/FairThreads/FTJava/DemosFairThreads

```
1  class Ball {
2    double x, y, angleX = 0, angleY = 0;
3    int radius = 8, steps = 100, scale = 5;
4    Color color;
5
6    public Ball(int x,int y,Color color){
7      this.x = x; this.y = y; this.color = color;
8    }
9    public Ball(int x,int y){
10     this(x,y,ColorBall.nextColor()); }
11   public void paint(Graphics g){
12     g.setColor(color);
13     g.fillOval((int)x-radius, (int)y-radius,radius*2,↩
           radius*2);
14   }
15   public void sine(){
16     angleY += (2*Math.PI/steps);
17     y += scale*Math.sin(angleY);
18   }
19   public void cosine(){
20     angleX += (2*Math.PI/steps);
21     x += scale*Math.sin(angleX+Math.PI/2);
22   } }
```

Applet et Fair Thread : classes Sin et Updating (2)

```
1  class Sin extends FairThread {
2    Ball ball;
3
4    public Move(Ball b){ ball = b; }
5    public void run(FairScheduler scheduler){
6      while(true){
7        ball.sine();
8        cooperate();
9      }
10   }
11 }
12
13 class Updating implements Fair {
14   Applet applet;
15
16   public Updating(Applet a){ applet = a; }
17   public void run(FairScheduler scheduler,FairThread ↩
         thread){
18     while(true){
19       applet.paint(applet.getGraphics());
20       thread.cooperate();
21     }
22   }
23 }
```

Applet et Fair Thread : classe Circle (3)

```
1  public class Circle extends Applet {
2    FairScheduler scheduler = new FairScheduler();
3    Ball ball = new Ball(startx,starty);
4
5    public void paint(Graphics g) {
6      super.paint(g);
7      ball.paint(g);
8      ball.color = ColorBall.nextColor();
9    }
10   void figure(Ball ball){
11     new Sin(ball).start(scheduler);
12     new Cos(ball).start(scheduler);
13   }
14   public void init(){
15     new FairThread(new Updating(this)).start(scheduler);
16     new FairThread(){
17       public void run(FairScheduler scheduler){
18         figure(ball);
19       }
20     }.start(scheduler);
21   }
22 }
```

Applet et Fair Thread : classe Circle (4)

Ajout facile d'autres fairthreads :

```
1 public class Lissajous extends Circle {
2   void figure(Ball ball){
3     super.figure(ball);
4     new Cos(ball).start(scheduler);
5   }
6 }
```

Modèle à mémoire distincte

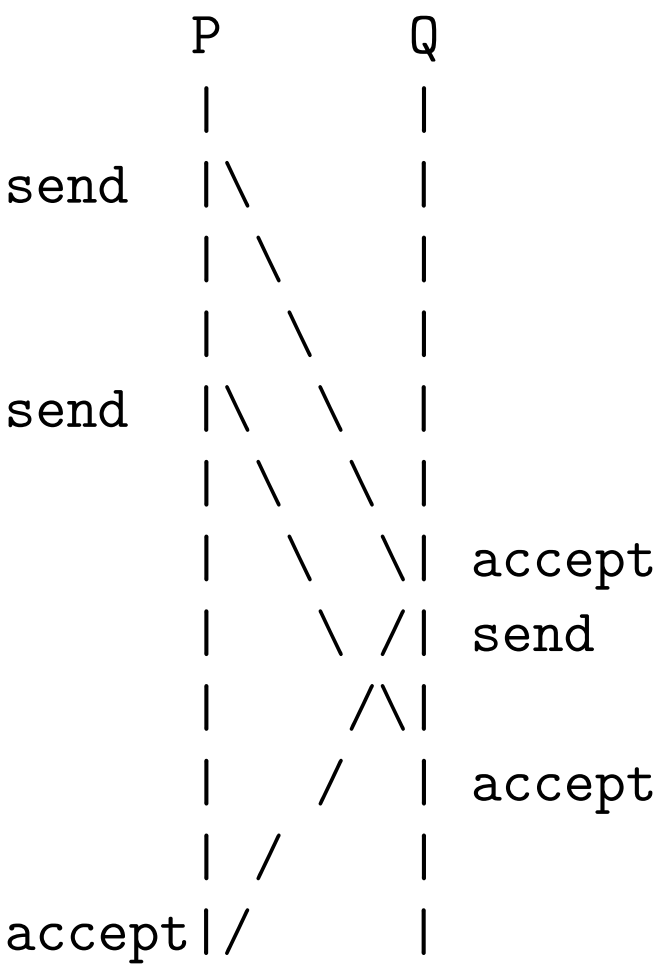
modèle à communication de messages (message passing)
2 primitives :

- “envoi un message” :
- “accepte un message”

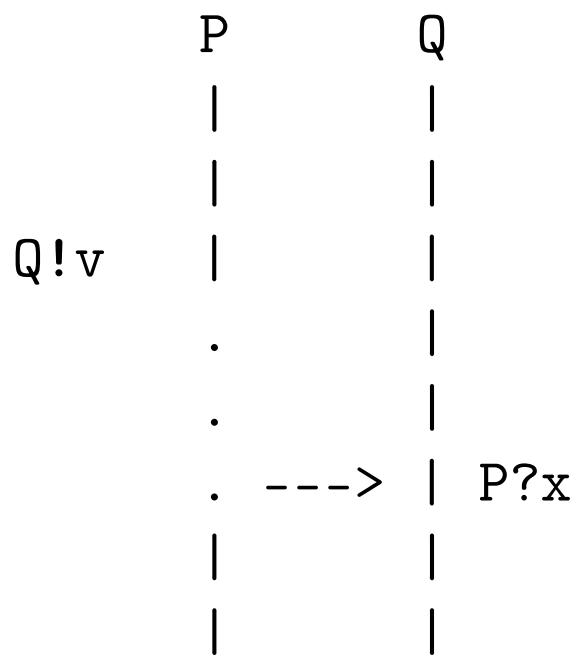
Caractéristiques

- envoi bloquant ou non
- réception bloquante ou non (*polling*)

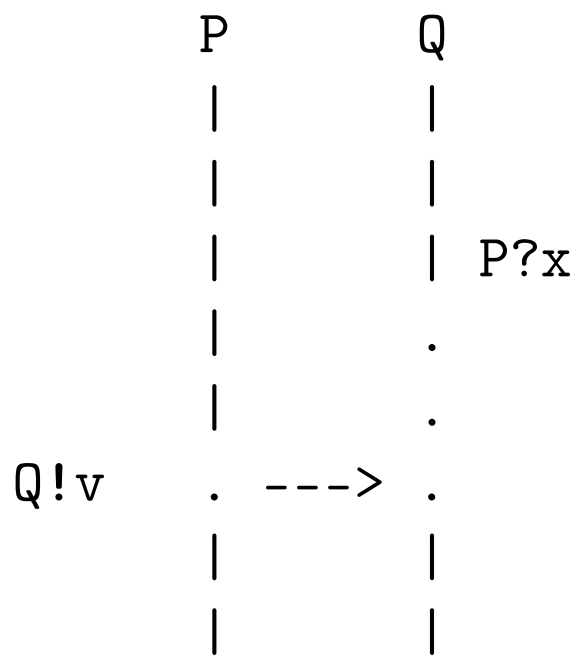
Communications asynchrones



Communications synchrones



Communications synchrones



Module Event - O'CAML

- communication synchrone
- canaux fortement typés
- si synchronisation, réception bloquante ou non (*poll*)

event.mli

```
 1  type 'a channel
 2  val new_channel: unit -> 'a channel
 3  type 'a event
 4  val send: 'a channel -> 'a -> unit event
 5  val receive: 'a channel -> 'a event
 6  val always: 'a -> 'a event
 7  val choose: 'a event list -> 'a event
 8  val wrap: 'a event -> f:('a -> 'b) -> 'b event
 9  val guard: (unit -> 'a event) -> 'a event
10  val sync: 'a event -> 'a
11  val select: 'a event list -> 'a
12  val poll: 'a event -> 'a option
```

Evénements, canaux et communication

- 2 types abstraits : `'a channel` et `'a event`
- `new_channel : unit -> 'a channel` : création d'un canal
- `send : 'a channel -> 'a -> unit event` : envoi une valeur `v` de type `'a` sur un canal `c` de type `'a channel`, retourne un événement dont la valeur est de type `unit` (valeur `() `)
- `receive : 'a channel -> 'a event`, retourne un événement de la valeur transmise.

`send` et `receive` ne sont pas bloquantes!!!

Synchronisation

- `sync : 'a event -> 'a` : fonction principale de synchronisation

  transforme un événement lié à une valeur en cette valeur.

Exemple 1 : partage de référence

```
1  let ch = Event.new_channel () ;;
2  let v = ref 0;;
3
4  let reader () = Event.sync (Event.receive ch);;
5  let writer () = Event.sync (Event.send ch ("S" ^ (↩
       string_of_int !v)));;
6
7  let loop_reader s d () =
8    for i=1 to 10 do
9      let r = reader() in
10       print_string (s ^ " " ^ r); print_newline();
11     Thread.delay  d
12   done ;;
13
14 let loop_writer d () =
15    for i=1 to 10 do incr v; writer(); Thread.delay d
16    done ;;
17
18 Thread.create (loop_reader "A" 1.1) ();;
19 Thread.create (loop_reader "B" 1.5) ();;
20 Thread.create (loop_reader "C" 1.9) ();;
21 Thread.delay 2.0;;
22 loop_writer 1. ();;
```

Exemple 1 : trace

```
% ocamlc -thread unix.cma threads.cma es1.ml
```

```
% ./a.out
C S1
A S2
B S3
C S4
A S5
B S6
C S7
A S8
B S9
C S10
```

```
% ./a.out
B S1
A S2
C S3
B S4
A S5
C S6
B S7
A S8
C S9
B S10
```

Exemple 2 : gensym (sans synchro)

```
 1  type uid = UID of string Event.channel;;
 2
 3  let makeUidSrc () =
 4    let ch = Event.new_channel () in
 5    let rec loop i = begin
 6       Event.send ch ("S"^(string_of_int i));
 7       loop (i+1)
 8    end in
 9      Thread.create (fun () -> loop 0) () ;
10      UID ch
11  ;;
12
13  let getUid (UID ch) = Event.receive ch;;
```

Exemple 2 : gensym (avec synchro)

```
 1  type uid = UID of string Event.channel;;
 2
 3  let makeUidSrc () =
 4    let ch = Event.new_channel () in
 5    let rec loop i = begin
 6       Event.sync (Event.send ch ("S"^(string_of_int i)));
 7       loop (i+1)
 8    end in
 9      Thread.create (fun () -> loop 0) () ;
10      UID ch
11  ;;
12
13  let getUid (UID ch) = Event.sync(Event.receive ch);;
```

Programme principal

```
1  let ch1 = makeUidSrc ();;
2
3  let main ti msg  () =
4    while (true) do
5      Thread.delay(ti);
6      let r = getUid ch1 in
7      print_string (msg); print_string " -- ";
8      print_string r;  print_newline();
9   done;;
10
11 Thread.create (main 1.1 "A") ();;
12
13 main 2.1 "B"  ();;
```

Trace

```
A -- S0
Src0
B -- S1
Src1
A -- S2
Src2
A -- S3
Src3
B -- S4
Src4
A -- S5
Src5
A -- S6
Src6
B -- S7
Src7
```

Polling

- `'a event -> 'a option` : version non bloquante de `sync`

  retourne `Some v` si un événement est présent, sinon `None`

Autres fonctions sur les événements

- `always : 'a -> 'a event` : crée un événement toujours prêt pour la synchronisation;
- `wrap : 'a event -> ('a -> 'b) -> 'b event` applique une fonction sur la valeur de l'événement (fonction de post-processing)
- `wrap_abort : 'a event -> (unit -> unit) -> 'a event` applique la fonction en cas de non sélection de l'événement

Choix d'un événement dans une liste

- `choose : 'a event list -> 'a event`
- `select : 'a event list -> 'a`

```
1       let select x = sync(choose x);;
```

Exemple : accumulateur +/-

3 canaux : addCh, SubCh et readCh :

```
1  let rec accum sum =
2     print_int sum; print_newline();
3     Event.sync  (
4       Event.choose [
5           wrap (receive addCh) (fun x -> accum(sum + x));
6           wrap (receive subCh) ( fun x -> accum(sum - x));
7           wrap (send  readCh sum)  ( fun x -> accum(sum))
8        ]
9     );;
```

wrap associe des actions aux communications!!!

Requêtes

```
1 let clientCallEvt x =
2   wrap (send reqCh x) (fun () -> receive replyCh);;
```

Mémoire partagée synchronisée (1)

M-variable :

- une M-variable est soit vide, soit pleine
- opération `take` : prendre la valeur d'une M-variable si elle est pleine, bloquante sinon
- opération `put` : remplit une M-variable, provoque une erreur si elle est pleine

Interface

```
1 type 'a mvar
2 val mVar : unit -> 'a mvar
3 exception Put
4 val mTake : 'a mvar -> 'a Event.event
5 val mPut : 'a mvar -> 'a -> unit
```

Une M-variable est construite dans un état vide.

Mémoire partagée synchronisée (2)

```
1  type 'a mvar = MV of ('a Event.channel * 'a Event.channel
2                        * bool Event.↩
                           channel);;
3
4  let mVar () =
5    let takeCh = Event.new_channel ()
6    and putCh = Event.new_channel ()
7    and ackCh = Event.new_channel () in
8    let rec empty () =
9      let x = Event.sync (Event.receive putCh) in
10       Event.sync (Event.send ackCh true);
11       full x
12   and full x =
13     Event.select
14       [Event.wrap (Event.send takeCh x) empty ;
15        Event.wrap (Event.receive putCh)
16                   (fun _ -> (Event.sync (Event.send  ackCh↩
                                false); full x))]
17   in
18     ignore (Thread.create empty ());
19     MV (takeCh, putCh, ackCh) ;;
```

Mémoire partagée synchronisée (3)

```
1
2  let mTake ( mv : 'a mvar) = match mv with
3     MV (takechannel, _, _ ) -> Event.receive takechannel ;;
4
5  exception Put;;
6  let mPut mv x = match mv with
7    MV (takechannel, putchannel, ackchannel) ->
8       Event.sync (Event.send putchannel x);
9       if (Event.sync( Event.receive ackchannel)) then ()
10      else raise Put ;;
```