CHOOE
Un gestionnaire d'environnement distribué

Franck LEBASTARD
CERMICS
INRIA
06902 Sophia-Antipolis Cedex
France

Résumé

CHOOE est un gestionnaire d'environnement distribué qui permet, d'une part, de facilement définir et faire évoluer dynamiquement un réseau de processus, et d'autre part, de gérer la communication entre ses éléments. Cette communication se fait par envois de messages en mode synchrone, asynchrone ou différé vers des destinataires qui sont soit des processus identifiés, soit des ensembles dynamiques de processus. En particulier, chaque élément du réseau peut à tout moment se déclarer compétent (ou incompétent) pour un ensemble de services, et toute communication adressée à un tel service atteint tous les processus compétents pour ce service au moment de la communication. Il est également possible de définir dynamiquement des liens de dépendance entre services, ce qui permet d'étendre virtuellement les domaines de compétence des processus.

CHOOE
A distributed environment manager

Abstract

CHOOE is a distributed environment manager that allows to define easily a process net and make it evolving dynamically. It also manages communication between processes. Messages can be sent either synchronously, asynchronously or in a "differed" mode, to receivers that can be identified processes or dynamic sets of processes. In particular, every element in the net can declare at any time to be competent (or not) for services, and a communication sent to any of these services only joins well-concerned processes. It is also possible to define dynamically dependencies between services and, in that way, to extend virtually process competences.

1 Description de CHOOE

CHOOE permet de construire et gérer un environnement distribué formé d'un ensemble de processus qui communiquent entre eux selon leurs besoins. L'ensemble des processus peut continuer d'évoluer sans aucune limitation alors même que la communication entre les éléments a déjà commencé.

1.1 Définition du réseau de processus

Le réseau de processus est initialisé en même temps que son premier élément est déclaré. Ajouter un processus au réseau se traduit ensuite par une simple déclaration d'existence à ce premier élément.

Le premier élément du réseau est donc amené à jouer un rôle particulier. Il est appelé **processus principal**. Sa spécificité de connaître nécessairement tous les nœuds du réseau et de disposer en permanence d'informations à jour à leur sujet, lui permet de répondre aux questions que peuvent se poser les processus sur leurs partenaires. Par exemple, quand l'un d'eux souhaite engager une communication vers un autre qu'il ne connaît pas encore, il se renseigne d'abord à son sujet auprès du processus principal. Une fois l'interlocuteur identifié, il établit le contact; il pourra directement s'adresser à lui pour toute communication ultérieure éventuelle. Nous verrons par la suite que le processus principal a également d'autres fonctions.

Concrètement, l'ajout au réseau d'un nouveau processus peut s'effectuer en CHOOE de deux façons différentes :

- par création provoquée par l'un des éléments du réseau;
- par déclaration d'un processus existant.

Chaque élément du réseau est par défaut identifié par un numéro qui lui est attribué lors de sa prise en compte. Le processus principal porte un numéro arbitraire[1] et chaque nouvel élément se voit attribuer le numéro suivant (soit `(numéro_principal + n)`, `n` étant le nombre de processus déjà présents dans le réseau).

Il peut également leur être attribué un nom. Ces noms permettent ensuite de les identifier symboliquement dans les programmes écrits au dessus de CHOOE.

La figure 1 présente un exemple de construction de réseau de processus. Cet exemple illustre le fait que déclarations et créations peuvent être simultanément utilisés pour

1. Choisi par l'utilisateur.

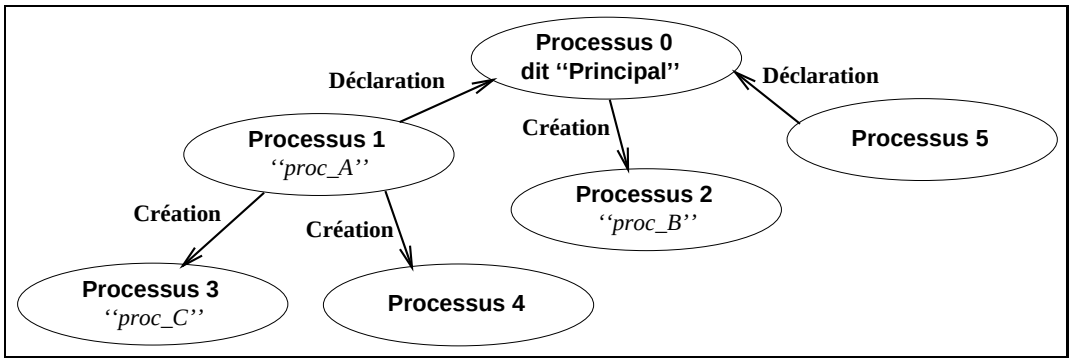

FIG. 1 - *Exemple de formation d'un environnement distribué*

construire un même réseau et que le nommage d'un processus est toujours possible sans être imposé.

Le processus principal a été le premier à être défini et initialisé; il porte ici le numéro 0. Un second processus s'est ensuite déclaré à lui sous le nom de "proc_A", il lui a été attribué le numéro 1. Lors de l'étape suivante, le processus principal a déclenché la création d'un nouveau processus qu'il a nommé "proc_B". Il lui a attribué le numéro suivant, ici le 3. Ensuite, "proc_A" a lui-même créé deux nouveaux processus 3 et 4, le premier nommé "proc_C", le second non nommé. Enfin, un dernier s'est déclaré sans se nommer au processus principal, lequel lui a attribué le numéro 5.

Une fois qu'un processus est élément d'un réseau, toute communication avec ses partenaires lui est possible : le graphe de communication est complet (figure 2). On peut ajouter que le réseau de processus peut continuer d'évoluer dynamiquement alors que le système a commencé à fonctionner et que des messages circulent sur le réseau : l'ajout et le retrait d'éléments sont immédiatement pris en compte.

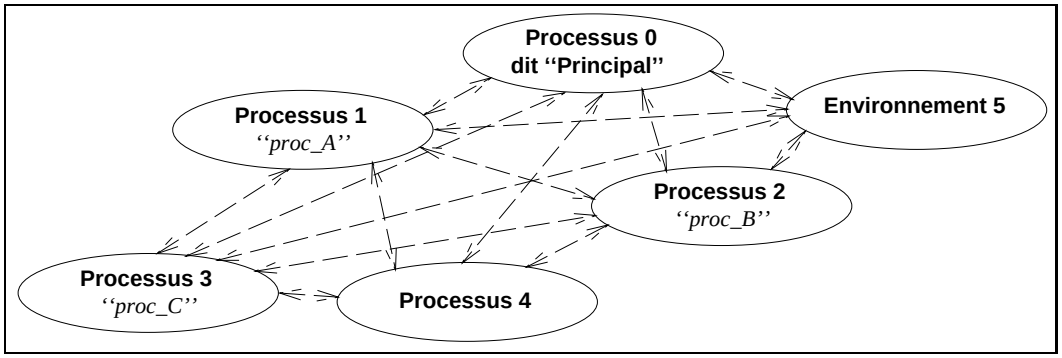

FIG. 2 - *Complétude du graphe de communication entre processus*

1.2 Les services

Chaque processus peut se déclarer compétent pour un ensemble de services. Dès qu'une déclaration de compétence[2] est faite, elle est immédiatement prise en compte et son auteur devient receveur de tout message adressé au service en question.

L'ensemble des compétences de chaque processus peut évoluer dynamiquement en cours de fonctionnement du système, aussi bien en ajout qu'en retrait. À tout moment, le processus principal connaît les compétences de chacun. Quand un processus souhaite envoyer un message à un service, il interroge d'abord le processus principal pour connaître la liste des fournisseurs; ce n'est que dans un deuxième temps qu'il envoie à chacun d'eux une copie de la requête. Cette mécanique interne est bien sûr cachée à l'utilisateur. Du fait de sa connaissance des compétences de chacun, le processus principal joue un rôle similaire au *répertoire* de [Djo93].

En CHOOE, un service est représenté par son nom. L'espace de nommage des services et des processus étant unique, on ne pourra pas utiliser comme identificateur de compétence un nom qui a déjà été utilisé pour identifier un processus, et inversement.

Il peut être intéressant que les prestataires d'un service donné reçoivent également les requêtes normalement adressées à un autre service. Cela peut permettre de traiter convenablement des commandes en principe destinées aux prestataires d'un service donné, mais pouvant en réalité être satisfaites par un public plus large. Si on considère par exemple les services `database` et `objectdatabase`, proposés par deux processus différents au sein d'un même réseau, on peut souhaiter que tout message adressé à `database` soit également envoyé à `objectdatabase`.

CHOOE permet de définir de tels liens de dépendance entre services. Il est ainsi possible de définir des hiérarchies, voire des treillis de services. L'analogie entre service et classe peut être faite, et le lien de dépendance peut également être interprété comme un lien d'héritage.

1.3 Gestion de la communication

Une unique commande `chooe-remote-call` permet d'adresser des messages aux différents processus du réseau. Cette commande prend trois arguments. Le premier identifie le destinataire du message, le second indique dans quel mode s'effectue la communication et le troisième est le message proprement dit.

Le destinataire détermine l'ensemble des processus qui vont recevoir le message. Il peut prendre différentes formes; il peut être :

- un nom ou un numéro de processus;
- un nom de service;

2. Compétence et service sont ici synonymes.

- un ensemble de numéros de processus et/ou de noms de processus ou de service;
- le mot réservé `all`. Tous les éléments du réseau reçoivent le message, y compris l'émetteur;
- une expression de sélection. Ne reçoivent le message que les processus qui vérifient la condition exprimée par l'expression de sélection.

Le second argument indique dans quel mode s'effectue la communication. Trois modes sont proposés :

- le mode **synchrone**. Le processus reste en attente jusqu'à ce qu'il ait obtenu les réponses de tous les processus qu'il a contactés. La fonction `chooe-remote-call` retourne alors (par défaut) l'ensemble des résultats rendus par ces processus. Si l'utilisateur le souhaite (positionnement d'un indicateur), il peut lui être précisé de quels processus émanent les différents résultats.

  L'inconvénient de ce mode est qu'il verrouille le processus en attente des résultats; il est alors sourd aux éventuelles requêtes des autres et il peut donc se produire des problèmes de **deadlock**. Cependant, tout deadlock immédiat, dû au fait que le destinataire comprend explicitement ou implicitement le processus émetteur, est géré par CHOOE et évité. C'est le cas quand le message est adressé à lui-même, à un service dont il est prestataire, au mot réservé `all` ou à une expression de sélection.

- le mode **asynchrone**. Dès que le message est émis, le processus émetteur reprend l'exécution en cours car aucune réponse n'est attendue. Dans ce mode, aucun problème de deadlock n'est à craindre.

- le mode **différé**. Comme dans le mode asynchrone, le processus émetteur reprend l'exécution en cours dès que le message est émis. Cependant, ici, des résultats sont attendus; le processus les recevra plus tard, de façon **différée**, quand les destinataires auront déterminé la réponse au message.

  En mode différé,

  - `chooe-remote-call` retourne un identificateur de message, lequel permettra de savoir quelles réponses correspondent à quels messages.
  - Le processus émetteur accède ensuite aux réponses aux requêtes au travers d'une boîte-aux-lettres.
  - L'identificateur de message permet de ne consulter que les réponses à un message donné.
  - Deux modes de lecture de la boîte-aux-lettres sont disponibles :
    - La simple consultation qui est non bloquante. S'il n'y a pas encore de réponse, CHOOE retourne un pointeur vide
    - L'attente de réponse. Ce second mode est bloquant mais il ne verrouille pas le processus qui peut recevoir, traiter et répondre aux éventuels messages émanant des autres processus.

– Ici également, si l'utilisateur le souhaite (positionnement du même indicateur qu'en mode synchrone), il peut lui être précisé de quels processus émanent les différents résultats qu'il trouve dans son courrier.

1.4 Interruption d'exécution

CHOOE permet à un processus membre du réseau d'interrompre l'exécution d'un autre, pour le libérer d'une tâche devenue inutile à effectuer.

Cette fonctionnalité permet, par exemple, une mise en concurrence efficace de processus travaillant à la réalisation d'une même tâche par des méthodes différentes. Quand le plus rapide a terminé et rendu son résultat, les autres peuvent être interrompus et ainsi redevenir disponibles.

1.5 Implantation

Une première implantation de CHOOE a été réalisée en Le_Lisp v15.25 [Cha89] au dessus de RPC-LINK [rpc91] de la société Ilog.

Elle permet de faire communiquer des processus UNIX pouvant être répartis sur différentes stations de travail reliées en réseau, par exemple par Internet.

Bien que particulièrement adaptée à faire communiquer des processus Le_Lisp, elle autorise[3] aussi la présence de processus C au sein des graphes de processus qu'elle gère.

La documentation présentée ci-après est le manuel de référence de cette implantation.

3. Parce qu'elle utilise le protocole RPC.

2 Manuel de référence de CHOOE (Le_Lisp)

Ce document présente l'interface fonctionnelle de CHOOE (Le_Lisp) version 1.06. Cette implantation de CHOOE a été réalisée en Le_Lisp v15.25 [Cha89] au dessus de RPC-LINK [rpc91] de la société Ilog.

- Sont présentées dans une première partie toutes les fonctions de construction de l'environnement distribué.
- Sont ensuite présentées toutes les primitives de gestion de la communication, puis des services.
- L'index des erreurs CHOOE est donné en §2.2.
- Enfin, l'index des fonctions et variables est donné en §2.3.

2.1 L'interface fonctionnelle de CHOOE

Ce document présente CHOOE (Le_Lisp) version 1.06 et son interface fonctionnelle. Cette implantation de CHOOE a été réalisée en Le_Lisp v15.25 [Cha89]; elle nécessite le chargement préalable du logiciel RPC-LINK [rpc91] de la société Ilog.

CHOOE **[FEATURE]**

Ce trait indique que le module CHOOE est chargé en mémoire. Si ce n'est pas le cas, on le charge par `^Lchooe`. Le fichier `chooe.ll` doit se trouver dans le répertoire courant.

Exemple :

```
? (FEATUREP 'chooe)
= ()
? ^Lchooe
Chargement de RPC...
Chargement de RPC Ok
Chargement de CHOOE...
CHOOE v1.06
Chargement de CHOOE Ok
[Variable chooe-main-process-host : crazy]
[Variable chooe-main-process-number : 0]
= chooe.ll
? (FEATUREP 'chooe)
= chooe
```

2.1.1 Gestion des processus éléments

CHOOE permet de gérer un environnement distribué formé d'un réseau de processus. L'environnement distribué comprend nécessairement un processus dit *principal*, dont la seule spécificité est de connaître tous les éléments composant l'environnement distribué. Grâce à lui, les communications à destinataire ambigu sont gérées efficacement (cf. §2.1.2).

À chaque processus est affecté un numéro qui l'identifie dans l'environnement distribué. Le processus principal porte le numéro contenu dans la variable `chooe-main-process-number` (`0` par défaut).

Au sein de l'environnement distribué, les processus sont représentés par des objets de type `Chooe-process`. L'affichage (le "`prin`") d'un tel objet est du type :

"`P[proc-id:host-id]*`"

`proc-id` est le nom donné par l'utilisateur au processus, ou, si aucun nom n'a été précisé, son numéro dans l'environnement distribué. `host-id` précise la machine sur laquelle se trouve le processus, en général son nom. Enfin, la présence d'une étoile (`*`) indique que l'objet représente le processus local, celui dans lequel il se trouve.

Construction et gestion du réseau de processus

Les fonctions suivantes permettent de construire et de gérer un environnement distribué.

(CHOOE-INIT-MAIN-PROCESS <procname>) [SUBR à 0 ou 1 argument]

Initialise le processus courant comme étant le processus principal de l'environnement distribué, et lui associe le nom <procname> si ce dernier est précisé.

Crée et retourne en valeur un objet `Chooe-process` le représentant.

Ce processus doit se trouver sur la machine dont le nom est précisé dans la variable système CHOOE-MAIN-PROCESS-HOST. Il portera dans l'environnement distribué le numéro contenu dans l'autre variable système CHOOE-MAIN-PROCESS-NUMBER (0 par défaut). Ces deux variables devront avoir même valeur dans tous les processus d'un même environnement distribué.

Lors de chaque création de processus, le processus principal sera consulté et devra affecter au nouveau un numéro identificateur. Ce numéro aura pour valeur CHOOE-MAIN-PROCESS-NUMBER + nombre de processus éléments de l'environnement distribué.

Exemple:

```
? (CHOOE-INIT-MAIN-PROCESS 'main)
= P[main:crazy]*
```

(CHOOE-DECLARE-PROCESS <procname>) [SUBR à 0 ou 1 argument]

Déclare le processus courant comme processus élément de l'environnement distribué et lui associe le nom <procname> si ce dernier est précisé.

Crée et retourne en valeur un objet `Chooe-process` le représentant.

REMARQUE : Toute déclaration de processus est automatiquement et immédiatement prise en compte dans l'environnement distribué. Dès lors, des messages peuvent lui être adressés.

Exemple (dans un processus CHOOE/Le_Lisp sur la machine `happy`) :

```
? (CHOOE-DECLARE-PROCESS 'elem1)
= P[elem1:happy]*
```

(CHOOE-CREATE-PROCESS <procname> <hostname> <command-line>) **[SUBR à 3 arguments]**

Permet, depuis n'importe quel processus élément du réseau, de créer un nouveau processus élément. Retourne en valeur un objet `Chooe-process` représentant ce nouveau processus dans le processus local.

<command-line> est exécutée sur la machine <hostname>. Cette commande UNIX doit provoquer la création d'un processus comprenant CHOOE, et exécuter, dans ce nouveau processus, la fonction CHOOE-INIT-LOCAL-PROCESS. CHOOE-INIT-LOCAL-PROCESS préviendra le processus créateur (celui où CHOOE-CREATE-PROCESS a été exécutée) quand l'opération aura été menée à bien.

Après avoir reçu la confirmation de création de CHOOE-INIT-LOCAL-PROCESS, crée et retourne en valeur un objet `Chooe-process` représentant le nouveau processus pris en compte dans le réseau.

REMARQUE : Toute création de processus est automatiquement et immédiatement prise en compte dans l'environnement distribué. Dès lors, des messages peuvent lui être adressés.

Exemple (dans le processus principal) :

```
? (CHOOE-CREATE-PROCESS 'elem2 'elissa "aida/chooe &")
= P[elem2:elissa]
```

(CHOOE-INIT-LOCAL-PROCESS) **[SUBR sans argument]**

Initialise le processus local comme processus élément de l'environnement distribué.

Retourne en valeur un objet `Chooe-process` le représentant.

REMARQUE : L'initialisation du processus principal doit, pour sa part, obligatoirement être effectuée au moyen de la primitive CHOOE-INIT-MAIN-PROCESS.

Exemple (dans un processus CHOOE/Le_Lisp sur la machine `elissa`):

```
? (CHOOE-INIT-LOCAL-PROCESS)
= P[elem2:elissa]*
```

(CHOOE-LOCAL-PROCESS-NUMBER) **[SUBR sans argument]**

Retourne le numéro du processus local.

Exemple (dans le processus principal) :

```
? (CHOOE-LOCAL-PROCESS-NUMBER)
= 0
```

(CHOOE-LOCAL-PROCESS-NAME) **[SUBR sans argument]**

Retourne le nom du processus local.

Exemple (dans le processus principal) :

```
? (CHOOE-LOCAL-PROCESS-NAME)
= main
```

(CHOOE-MAIN-PROCESS-P) **[SUBR sans argument]**

Prédicat indiquant si le processus local est le processus principal.

Exemple (dans le processus principal) :

```
? (CHOOE-MAIN-PROCESS-P)
= t
```

2.1.2 Gestion des communications

CHOOE gère la communication entre les processus éléments du réseau au moyen de la fonction CHOOE-REMOTE-CALL, décrite ci-après :

(CHOOE-REMOTE-CALL <receiver> <mode> <command>)
[SUBR à 3 arguments]

Envoie la commande <command> au destinataire <receiver> (qui par défaut l'évalue) dans le mode <mode>. En Le_Lisp, la commande doit être une expression Lisp "bien formée" ne comportant pas de structure circulaire.

Le traitement de la commande est réalisée par la fonction CHOOE-COMMAND-EVAL qui est appelée avec en argument le "message réseau intégral" (voir la description de cette fonction). En cas d'erreur dans l'évaluation de la commande, CHOOE-COMMAND-ERROR est appelée avec également en argument le message réseau intégral.

Le destinataire peut être :

- un identificateur de processus, i.e. un nom (symbole) ou un numéro;
- un identificateur de service (symbole) (cf. §2.1.3);
- une liste composée d'identificateurs de processus et/ou de services;
- un objet ou une liste d'objets de type `Chooe-process`;
- le mot-clé `all` : tous les processus éléments du réseau recevront le message, y compris le processus émetteur;
- un sélecteur qui détermine dynamiquement quels processus reçoivent le message. Le sélecteur est décrit sous forme d'une liste à deux éléments, le premier étant le mot-clé `selector`, le second une lambda-expression à un argument. Cette lambda est évaluée dans chaque processus élément en liant à l'argument le message intégral transmis. Si l'évaluation de la lambda retourne un résultat vrai au sens lisp, le message est traité par le processus, pas dans le cas contraire.

Trois modes sont disponibles :

- `async`. Le message est transmis en mode asynchrone et CHOOE-REMOTE-CALL rend la main tout-de-suite. Aucun résultat n'est attendu : la fonction retourne `t`.
- `sync`. Le message est transmis en mode synchrone. CHOOE-REMOTE-CALL reste en attente et retourne la liste du ou des résultats renvoyés par les processus interrogés. En effet, quel que soit le destinataire, la réponse est toujours retournée sous forme d'une liste, ne comprenant éventuellement qu'une seule réponse (cas d'une requête adressée à un processus particulier).
  Quand la variable système CHOOE-SENDERS-SPECIFIED-P est positionnée à `t` (la valeur par défaut est `()`), la valeur retournée par la fonction CHOOE-REMOTE-CALL est structurée sous forme d'une a-liste où les clés sont les identificateurs (nom, sinon numéro) des processus ayant répondu, et les valeurs sont les réponses qu'ils ont retournées.
- `deferred`. Le message est transmis en mode asynchrone et CHOOE-REMOTE-CALL rend la main tout-de-suite. Cependant, un ou des résultats sont attendus : ils arrivent de façon différée et sont immédiatement placés dans une boîte-aux-lettres. Il peuvent alors être exploités dès que nécessaire.

Dans ce mode, CHOOE-REMOTE-CALL retourne un symbole qui est l'identificateur de la requête envoyée. Il est utilisé avec les fonctions CHOOE-READ-MAIL, CHOOE-WAIT-FOR-MAIL et CHOOE-CONSULT-MAIL pour interroger la boîte-aux-lettres et récupérer la ou les réponses à la requête correspondante.

REMARQUE : Quand la variable CHOOE-MESSAGES-P vaut `t` (valeur par défaut), l'arrivée d'un résultat dans la boîte-aux-lettres est signalée par le message "New mail".

Exemple (dans le processus "`elem2`"):

```
? (CHOOE-REMOTE-CALL 'elem1 'async
?         '(PRINT "Received in" (CHOOE-LOCAL-PROCESS-NUMBER)))
= t
? (CHOOE-REMOTE-CALL 'elem1 'sync
?         '(PRINT "Received in" (CHOOE-LOCAL-PROCESS-NUMBER)))
= (1)
? (CHOOE-REMOTE-CALL '(main 1 elem2) 'sync
?         '(PRINT "Received in" (CHOOE-LOCAL-PROCESS-NUMBER)))
Received in 2
= (0 1 2)
? (LET ((CHOOE-SENDERS-SPECIFIED-P t))
?      (CHOOE-REMOTE-CALL '(main 1 elem2) 'sync
?               '(PRINT "Received in" (CHOOE-LOCAL-PROCESS-NUMBER))))
Received in 2
= ((main . 0) (elem1 . 1) (elem2 . 2))
? (CHOOE-REMOTE-CALL 'all 'sync
?         '(PRINT "Received in" (CHOOE-LOCAL-PROCESS-NUMBER)))
Received in 2
= (2 0 1)
? (CHOOE-REMOTE-CALL
?      '(selector (LAMBDA (full-message)
?                     (ZEROP (MODULO (CHOOE-LOCAL-PROCESS-NUMBER)
?                                    2))))
?      'sync
?      '(PRINT "Received in" (CHOOE-LOCAL-PROCESS-NUMBER)))
Received in 2
= (2 0)
```

(CHOOE-COMMAND-EVAL <full-message>) **[SUBR à 1 argument]**

Fonction appelée par CHOOE-REMOTE-CALL pour traiter une commande. Elle prend en argument le message réseau intégral ("full message") reçu par le processus.

Ce dernier a comme structure :

```
(sender-number message-ident receiver mode command)
```

Par défaut, la définition de CHOOE-COMMAND-EVAL est :

```
(DE chooe-command-eval (full-message)
    (EVAL (CAR (CDDDDR full-message))))
```

L'utilisateur a tout loisir de la redéfinir pour ses besoins propres.

(CHOOE-COMMAND-ERROR <full-message>) **[SUBR à 1 argument]**

Fonction appelée en cas d'erreur dans l'évaluation de la commande. Elle reçoit en argument le message réseau intégral ("full message") reçu par le processus.

Par défaut, la définition de CHOOE-COMMAND-ERROR est :

```
(DE chooe-command-error (full-message)
    (PRINT "Error valuing command from "
           (CAR full-message) " : " (CAR (CDDDDR full-message)))
    `(chooe-command-error
              ,(chooe-local-process-number) ,full-message))
```

L'utilisateur a tout loisir de la redéfinir pour ses besoins propres.

(CHOOE-READ-MAIL <message-ident>) **[SUBR à 0 ou 1 argument]**

Si <message-ident> n'est pas précisé, retourne le contenu de la boîte-aux-lettres. L'information y est structurée sous forme d'une a-liste où les clés sont les identificateurs de requêtes et où les valeurs sont les réponses reçues pour chaque requête.

Si <message-ident> est précisé, CHOOE-READ-MAIL retourne uniquement les résultats reçus en réponse à la requête d'identificateur <message-ident>.

Quand la variable système CHOOE-SENDERS-SPECIFIED-P est positionnée à `t` (la valeur par défaut est `()`), la réponse à une requête est elle-même structurée sous forme d'une a-liste où les clés sont les identificateurs (nom, sinon numéro) des processus ayant répondu, et les valeurs sont les réponses qu'ils ont retournées.

Un résultat est retiré de la boîte-à-lettres dès qu'il a été lu.

Exemple:

```
? (CHOOE-REMOTE-CALL '(elem1 2) 'deferred
                     '(CHOOE-LOCAL-PROCESS-NUMBER))
```

```
= g123
? New mail
 New mail
? (CHOOE-READ-MAIL)
= ((g123 1 2))
? (CHOOE-READ-MAIL)
= ()
? (CHOOE-REMOTE-CALL '(elem1 2) 'deferred
                     '(CHOOE-LOCAL-PROCESS-NUMBER))
= g124
? New mail
 New mail
? (LET ((CHOOE-SENDERS-SPECIFIED-P t))
?      (CHOOE-READ-MAIL 'g124))
= ((elem1 . 1) (elem2 . 3))
? (CHOOE-READ-MAIL 'g124)
= ()
```

(CHOOE-WAIT-FOR-MAIL <message-ident>) [SUBR à 0 ou 1 argument]

Se met en attente d'arrivée d'un mail. Si <message-ident> n'est pas précisé, retourne le contenu de la boîte-aux-lettres dès l'arrivée d'un message (même structure de l'information que pour la fonction CHOOE-READ-MAIL).

Si <message-ident> est précisé, CHOOE-WAIT-FOR-MAIL ne rend la main qu'à réception de résultats reçus en réponse à la requête d'identificateur <message-ident>.

Un résultat est retiré de la boîte-à-lettres dès qu'il a été lu.

Exemple:

```
? (CHOOE-REMOTE-CALL '(main elem1 elem3) 'deferred
                     '(CHOOE-LOCAL-PROCESS-NUMBER))
= g125
? (CHOOE-WAIT-FOR-MAIL 'g125)
 New mail
= (0)
? New mail
 New mail
(LET ((CHOOE-SENDERS-SPECIFIED-P t))
?    (CHOOE-WAIT-FOR-MAIL 'g125))
= ((elem1 . 2) (elem2 . 3))
? (CHOOE-READ-MAIL 'g125)
= ()
```

(CHOOE-CONSULT-MAIL <message-ident>) [SUBR à 0 ou 1 argument]

Permet de consulter le contenu de la boîte-aux-lettres sans le modifier.

Si <message-ident> n'est pas précisé, retourne le contenu complet de la boîte-aux-lettres.

Si <message-ident> est précisé, CHOOE-CONSULT-MAIL retourne uniquement les résultats reçus en réponse à la requête d'identificateur <message-ident>.

Exemple:

```
? (CHOOE-REMOTE-CALL '(main elem1 elem2) 'deferred
                     '(CHOOE-LOCAL-PROCESS-NUMBER))
= g126
? New mail
 New mail
 New mail
? (CHOOE-CONSULT-MAIL)
= ((g126 0 1 2))
? (LET ((CHOOE-SENDERS-SPECIFIED-P t))
?      (CHOOE-CONSULT-MAIL))
= ((g126 (main . 0) (elem1 . 1) (elem2 . 2)))
? (CHOOE-CONSULT-MAIL 'g126)
= (0 1 2)
? (LET ((CHOOE-SENDERS-SPECIFIED-P t))
?      (CHOOE-CONSULT-MAIL 'g126))
= ((main . 0) (elem1 . 1) (elem2 . 2))
? (CHOOE-CONSULT-MAIL)
= ((g126 0 1 2))
```

2.1.3 Gestion des services

Chaque processus peut se déclarer compétent pour un ensemble de services donnés. Après l'avoir déclaré, il reçoit automatiquement toute requête adressée à l'un de ces services.

(CHOOE-ADD-SERVICE <service>) [SUBR à 1 argument]

Déclare le processus local "compétent" pour le service <service> (symbole). Toute requête adressée à ce service (i.e. à l'ensemble des processus compétents pour ce service) lui est dès lors également adressée.

Exemple (dans le processus "elem2"):

```
? (CHOOE-ADD-SERVICE 'database)
= database
? (CHOOE-REMOTE-CALL 'database 'sync
          '(PRINT "Received in" (CHOOE-LOCAL-PROCESS-NUMBER)))
Received in 2
= (2)
? ; Execute (CHOOE-ADD-SERVICE 'database) dans le processus elem1
? (CHOOE-REMOTE-CALL 'elem1 'async
                     '(CHOOE-ADD-SERVICE 'database))
= t
? (CHOOE-REMOTE-CALL 'database 'sync
          '(PRINT "Received in" (CHOOE-LOCAL-PROCESS-NUMBER)))
Received in 2
= (1 2)
```

(CHOOE-REMOVE-SERVICE <service>) [SUBR à 1 argument]

Retire au processus local la "compétence" pour le service <service>. Dès lors, toute requête adressée à ce service (i.e. à l'ensemble des processus compétents pour ce service) ne lui est plus adressée.

Exemple (dans le processus "elem2"):

```
? (CHOOE-REMOVE-SERVICE 'database)
= database
? (CHOOE-REMOTE-CALL 'database 'sync
          '(PRINT "Received in" (CHOOE-LOCAL-PROCESS-NUMBER)))
= (1)
```

(CHOOE-LOCAL-SERVICES) [SUBR sans argument]

Retourne la liste des services pour lesquels le processus local s'est déclaré compétent.

Exemple (dans le processus "elem1"):

```
? (CHOOE-LOCAL-SERVICES)
= (database)
```

(CHOOE-ALL-SERVICES) **[SUBR sans argument]**

Retourne la liste des services disponibles dans l'environnement distribué (dans l'ensemble des processus éléments du réseau).

Exemple (dans le processus "`elem2`"):

```
? (CHOOE-LOCAL-SERVICES)
= ()
? (CHOOE-ALL-SERVICES)
= (database)
```

(CHOOE-SERVICE-FURNISHERS <service>) **[SUBR à 1 argument]**

Retourne la liste des prestataires du service <service>. Elle est formée des identificateurs des processus correspondants (nom, sinon numéro).

Exemple :

```
? (CHOOE-SERVICE-FURNISHERS 'database)
= (elem1)
```

Il peut être intéressant que les prestataires d'un service donné reçoivent également les requêtes normalement adressées à un autre service (cf. §1.2). C'est ce que permettent les fonctions suivantes.

(CHOOE-DEFINE-SUBSERVICE <service> <subservice>) **[SUBR à 2 arguments]**

Définit le service <subservice> comme "sous-service" du service <service>. Suite à cette déclaration, toute requête adressée au service <service> sera également (et automatiquement) envoyée au service <subservice>.

Exemple (dans le processus "`elem2`"):

```
? (CHOOE-ADD-SERVICE 'objectdatabase)
= objectdatabase
? (CHOOE-REMOTE-CALL 'database 'sync
```

```
          '(PRINT "Received in" (CHOOE-LOCAL-PROCESS-NUMBER)))
= (1)
? (CHOOE-DEFINE-SUBSERVICE 'database 'objectdatabase)
= objectdatabase
? (CHOOE-REMOTE-CALL 'database 'sync
          '(PRINT "Received in" (CHOOE-LOCAL-PROCESS-NUMBER)))
Received in 2
= (1 2)
? (CHOOE-SERVICE-FURNISHERS 'database)
= (elem1 elem2)
```

(CHOOE-DROP-SUBSERVICE <service> <subservice>) **[SUBR à 2 arguments]**

Annule la commande précédente. Suite à cette déclaration, toute requête adressée au service <service> n'est plus envoyée au service <subservice>.

Exemple (dans le processus "elem2"):

```
? (CHOOE-DROP-SUBSERVICE 'database 'objectdatabase)
= t
? (CHOOE-REMOTE-CALL 'database 'sync
          '(PRINT "Received in" (CHOOE-LOCAL-PROCESS-NUMBER)))
= (1)
```

2.1.4 Commandes diverses

(CHOOE-INTERRUPT-PROCESS <procid>) **[SUBR à 1 argument]**

Interrompt l'exécution du processus d'identificateur (nom ou numéro) <procid>. Après l'interruption, le processus est à nouveau disponible pour recevoir de nouvelles requêtes et pour effectuer de nouvelles tâches.

REMARQUE : On interrompt le processus en lui envoyant le signal d'interruption (signal numéro 2). Le numéro du signal envoyé est contenu dans la variable CHOOE-INTERRUPT-SIGNAL. Tout type de signal peut donc être envoyé à un processus avec la commande CHOOE-INTERRUPT-PROCESS par simple modification de cette variable.

Exemple (dans le processus "elem2"):

```
? (CHOOE-INTERRUPT-PROCESS 'elem1)
= t
```

2.2 Messages d'erreurs et autres de CHOOE

Codes	Messages d'erreur et autres de CHOOE
chooerr01	"Environnement déjà initialisé"
	"Process already initialized"
chooerr02	"Mauvaise valeur de chooe-main-process-host"
	"Bad chooe-main-process-host value"
chooerr03	"La variable Unix HOSTNAME n'a pas été définie"
	"Unix variable HOSTNAME is undefined"
chooerr04	"Déjà un nom de processus ou de service"
	"Already a process name or a service name"
chooerr05	"Pas un nom de processus ou de service"
	"Not an existing service name or process name"
chooerr06	"Mauvais nom de processus ou de service"
	"Bad service name or process name"
chooerr07	"Mauvais numéro de processus local"
	"Bad local process number"
chooerr08	"Pas un numéro de processus existant"
	"Not an existing process number"
chooerr09	"Mauvais nom de processus"
	"Bad local process name"
chooerr10	"Pas un nom de processus existant"
	"Not an existing process name"
chooerr11	"Déjà un nom de processus"
	"Already a process name"
chooerr21	"Pas un mode de communication existant"
	"Not an existing communication mode"
chooerr22	"Pas un numéro de signal"
	"Not a signal number"
chooerr91	"Pas un symbole"
	"Not a symbol"
chooerr92	"Pas un symbole ou un entier"
	"Not a symbol or an integer"
chooerr93	"Mauvais accès a un champ"
	"Bad field access"

2.3 Index des fonctions et des variables

CHOOE [FEATURE] 9
CHOOE-INTERRUPT-SIGNAL [Variable] 20
CHOOE-MAIN-PROCESS-HOST [Variable] 10
CHOOE-MAIN-PROCESS-NUMBER [Variable] 10
CHOOE-MESSAGES-P [Variable] 14
CHOOE-SENDERS-SPECIFIED-P [Variable] 13, 15
(CHOOE-ADD-SERVICE <service>) [SUBR à 1 argument] 17
(CHOOE-ALL-SERVICES) [SUBR sans argument] 19
(CHOOE-COMMAND-ERROR <full-message>) [SUBR sans argument] 13, 15
(CHOOE-COMMAND-EVAL <full-message>) [SUBR sans argument] 13, 14
(CHOOE-CONSULT-MAIL <message-ident>) [SUBR à 0 ou 1 argument] 17
(CHOOE-CREATE-PROCESS <procname> <hostname> <command-line>)
[SUBR à 3 arguments] 11
(CHOOE-DECLARE-PROCESS <procname>) [SUBR à 0 ou 1 argument] 10
(CHOOE-DEFINE-SUBSERVICE <service> <subservice>) [SUBR à 2 arguments] 19
(CHOOE-DROP-SUBSERVICE <service> <subservice>) [SUBR à 2 arguments] 20
(CHOOE-INIT-LOCAL-PROCESS) [SUBR sans argument] 11
(CHOOE-INIT-MAIN-PROCESS <procname>) [SUBR à 0 ou 1 argument] 10
(CHOOE-INTERRUPT-PROCESS <procid>) [SUBR à 1 argument] 20
(CHOOE-LOCAL-PROCESS-NAME) [SUBR sans argument] 12
(CHOOE-LOCAL-PROCESS-NUMBER) [SUBR sans argument] 12
(CHOOE-LOCAL-SERVICES) [SUBR sans argument] 18
(CHOOE-MAIN-PROCESS-P) [SUBR sans argument] 12
(CHOOE-READ-MAIL <message-ident>) [SUBR à 0 ou 1 argument] 15
(CHOOE-REMOTE-CALL <receiver> <mode> <command>) [SUBR à 3 arguments] ... 13
(CHOOE-REMOVE-SERVICE <service>) [SUBR à 1 argument] 18
(CHOOE-SERVICE-FURNISHERS <service>) [SUBR à 1 argument] 19
(CHOOE-WAIT-FOR-MAIL <message-ident>) [SUBR à 0 ou 1 argument] 16

Références

[Cha89] J. Chailloux. *LE_LISP de l'INRIA Version 15.22, Manuel de référence*. INRIA, Le Chesnay (France), Janvier 1989. 430 pages.

[Djo93] C. Djossou. *Approche multi-processus pour la coopération entre systèmes à base de connaissances*. Thèse de doctorat, Université de Nice - Sophia-Antipolis, Février 1993. 125 pages.

[rpc91] Ilog, Gentilly, France. *RPC Link version 3.25.2, Manuel de référence*, 1991. 43 pages.

Table des matières

1 Description de CHOOE **1**
1.1 Définition du réseau de processus 1
1.2 Les services 3
1.3 Gestion de la communication 3
1.4 Interruption d'exécution 5
1.5 Implantation 5

2 Manuel de référence de CHOOE (Le_Lisp) **7**
2.1 L'interface fonctionnelle de CHOOE 9
2.1.1 Gestion des processus éléments 9
2.1.2 Gestion des communications 12
2.1.3 Gestion des services 17
2.1.4 Commandes diverses 20
2.2 Messages d'erreurs et autres de CHOOE 21
2.3 Index des fonctions et des variables 23