Sérialisation/Désérialisation

Lycée Thiers

Crédit

- ≪ Informatique MP2I et MPI - CPGE 1re et 2e années - Nouveaux programmes ≫ (Vincent Barra)

Crédit

- « Informatique MP2I et MPI - CPGE 1re et 2e années - Nouveaux programmes » (Vincent Barra)
- Wikipedia

Présentation

- La *sérialisation* (marshalling) est le codage d'une information sous la forme d'une suite d'informations plus petites (dites *atomiques*) pour, par exemple, sa sauvegarde (persistance) ou son transport sur le réseau (proxy, RPC...).

Présentation

- La *sérialisation* (marshalling) est le codage d'une information sous la forme d'une suite d'informations plus petites (dites *atomiques*) pour, par exemple, sa sauvegarde (persistance) ou son transport sur le réseau (proxy, RPC...).
- L'activité réciproque, visant à décoder cette suite pour créer une copie conforme de l'information d'origine, s'appelle la *désérialisation* (ou unmarshalling).

Difficultés

- Plus le mécanisme de sérialisation est spécialisé dans un type de données spécifiques plus il est performant.

Difficultés

- Plus le mécanisme de sérialisation est spécialisé dans un type de données spécifiques plus il est performant.
- Par exemple, si on veut coder des entiers de valeurs entre 0 et 255, il suffit de un octet par entier.

Difficultés

- Plus le mécanisme de sérialisation est spécialisé dans un type de données spécifiques plus il est performant.
- Par exemple, si on veut coder des entiers de valeurs entre 0 et 255, il suffit de un octet par entier.
- Si ce ne sont pas des entiers mais des objets complexes, il faut associer des informations permettant de coder le type précis de chaque objet.

Dilemne : Précision de l'information/Taille du fichier généré

- Il y a un choix à faire selon que l'on veut privilégier une désérialisation à l'identique du fichier sérialisé ou la place prise par ce fichier.

Dilemne : Précision de l'information/Taille du fichier généré

- Il y a un choix à faire selon que l'on veut privilégier une désérialisation à l'identique du fichier sérialisé ou la place prise par ce fichier.
- Exemple de la police de caractère dans un fichier PDF :

Dilemne : Précision de l'information/Taille du fichier généré

- Il y a un choix à faire selon que l'on veut privilégier une désérialisation à l'identique du fichier sérialisé ou la place prise par ce fichier.
- Exemple de la police de caractère dans un fichier PDF :
  - on peut transmettre la description complète du tracé des caractères,

Dilemne : Précision de l'information/Taille du fichier généré

- Il y a un choix à faire selon que l'on veut privilégier une désérialisation à l'identique du fichier sérialisé ou la place prise par ce fichier.
- Exemple de la police de caractère dans un fichier PDF :
  - on peut transmettre la description complète du tracé des caractères,
  - ou bien on peut indiquer seulement le nom de la police et quelques autres caractéristiques et laisser le soin à la machine distante de choisir elle-même la police la plus adaptée parmi celles dont elle dispose.

Informations non sérialisables

- Certaines informations ne sont pas sérialisables.

Informations non sérialisables

- Certaines informations ne sont pas sérialisables.
- Par exemple le descripteur de fichier : d'une machine à l'autre, ces descripteurs sont attribués de façon arbitraires par le système d'exploitation : sérialiser leur contenu n'a donc pas de sens.
  Il faut plutôt encoder des informations qui permettront de reconstruire le descripteur au moment de la désérialisation.

Informations non sérialisables

- Certaines informations ne sont pas sérialisables.
- Par exemple le descripteur de fichier : d'une machine à l'autre, ces descripteurs sont attribués de façon arbitraires par le système d'exploitation : sérialiser leur contenu n'a donc pas de sens.
  Il faut plutôt encoder des informations qui permettront de reconstruire le descripteur au moment de la désérialisation.
- Autre exemple : sérialisation des pointeurs. La valeur du pointeur (adresse pointée) dans un programme dépend du moment où est lancé ce programme.

Choix de l'encodage

encodage binaire : les fichiers binaires sont plus compacts, le code pour parser (analyser) ce type de données est plus simple à mettre en œuvre, la lecture et l'écriture sont moins exigeantes en ressources processeurs ;

Choix de l'encodage

encodage binaire : les fichiers binaires sont plus compacts, le code pour parser (analyser) ce type de données est plus simple à mettre en œuvre, la lecture et l'écriture sont moins exigeantes en ressources processeurs ;

encodage textuel les fichiers textes sont plus simples à vérifier ou modifier à la main, ils posent moins de problèmes de portabilité et sont plus simples à maintenir ou à faire évoluer.

Codage binaire

Contrainte de portabilité

- Si la machine distante utilise un autre processeur, elle doit pouvoir désérialiser un bloc de données en tenant compte :

Codage binaire

Contrainte de portabilité

- Si la machine distante utilise un autre processeur, elle doit pouvoir désérialiser un bloc de données en tenant compte :
  - des problèmes d'alignement : si on transmet un caractère (1 octet) puis un entier (4 octets), doit on compter en tout 4 octets en ≪ concaténant ≫ la fin du caractère avec le début de l'entier, ou au contraire prévoire 4 octets par données (et donc en utilisant inutilement 3 octets de trop pour le caractère)

Codage binaire

Contrainte de portabilité

- Si la machine distante utilise un autre processeur, elle doit pouvoir désérialiser un bloc de données en tenant compte :
  - des problèmes d'alignement : si on transmet un caractère (1 octet) puis un entier (4 octets), doit on compter en tout 4 octets en ≪ concaténant ≫ la fin du caractère avec le début de l'entier, ou au contraire prévoire 4 octets par données (et donc en utilisant inutilement 3 octets de trop pour le caractère)
  - ou d'endianness (boutisme) : où est le bit de poids fort ?

Codage binaire

Contrainte de portabilité

- Si la machine distante utilise un autre processeur, elle doit pouvoir désérialiser un bloc de données en tenant compte :
  - des problèmes d'alignement : si on transmet un caractère (1 octet) puis un entier (4 octets), doit on compter en tout 4 octets en ≪ concaténant ≫ la fin du caractère avec le début de l'entier, ou au contraire prévoire 4 octets par données (et donc en utilisant inutilement 3 octets de trop pour le caractère)
  - ou d'endianness (boutisme) : où est le bit de poids fort ?
- Il est utile d'utiliser des conventions. Par exemple :

Codage binaire

Contrainte de portabilité

- Si la machine distante utilise un autre processeur, elle doit pouvoir désérialiser un bloc de données en tenant compte :
  - des problèmes d'alignement : si on transmet un caractère (1 octet) puis un entier (4 octets), doit on compter en tout 4 octets en ≪ concaténant ≫ la fin du caractère avec le début de l'entier, ou au contraire prévoire 4 octets par données (et donc en utilisant inutilement 3 octets de trop pour le caractère)
  - ou d'endianness (boutisme) : où est le bit de poids fort ?
- Il est utile d'utiliser des conventions. Par exemple :
  - pas d'alignement,

Codage binaire

Contrainte de portabilité

- Si la machine distante utilise un autre processeur, elle doit pouvoir désérialiser un bloc de données en tenant compte :
  - des problèmes d'alignement : si on transmet un caractère (1 octet) puis un entier (4 octets), doit on compter en tout 4 octets en ≪ concaténant ≫ la fin du caractère avec le début de l'entier, ou au contraire prévoire 4 octets par données (et donc en utilisant inutilement 3 octets de trop pour le caractère)
  - ou d'endianness (boutisme) : où est le bit de poids fort ?
- Il est utile d'utiliser des conventions. Par exemple :
  - pas d'alignement,
  - encodage des types entiers C en fonction de leur empreinte mémoire ; tout en big-endian

Codage binaire

Contrainte de portabilité

- Si la machine distante utilise un autre processeur, elle doit pouvoir désérialiser un bloc de données en tenant compte :
  - des problèmes d'alignement : si on transmet un caractère (1 octet) puis un entier (4 octets), doit on compter en tout 4 octets en ≪ concaténant ≫ la fin du caractère avec le début de l'entier, ou au contraire prévoire 4 octets par données (et donc en utilisant inutilement 3 octets de trop pour le caractère)
  - ou d'endianness (boutisme) : où est le bit de poids fort ?
- Il est utile d'utiliser des conventions. Par exemple :
  - pas d'alignement,
  - encodage des types entiers C en fonction de leur empreinte mémoire ; tout en big-endian
  - les nombres à vigules flottantes sont codées selon la norme IEEE754

Codage binaire

Contrainte de portabilité

- Si la machine distante utilise un autre processeur, elle doit pouvoir désérialiser un bloc de données en tenant compte :
  - des problèmes d'alignement : si on transmet un caractère (1 octet) puis un entier (4 octets), doit on compter en tout 4 octets en ≪ concaténant ≫ la fin du caractère avec le début de l'entier, ou au contraire prévoire 4 octets par données (et donc en utilisant inutilement 3 octets de trop pour le caractère)
  - ou d'endianness (boutisme) : où est le bit de poids fort ?
- Il est utile d'utiliser des conventions. Par exemple :
  - pas d'alignement,
  - encodage des types entiers C en fonction de leur empreinte mémoire ; tout en big-endian
  - les nombres à vigules flottantes sont codées selon la norme IEEE754
- Exemples : protocoles GIOP de CORBA ; RMI de Java

Codage textuel

- Choisir un protocole pour séparer les champs, pour encoder des données binaires

Codage textuel

- Choisir un protocole pour séparer les champs, pour encoder des données binaires
- Dans le passé, on utilisait souvent un dérivé de XML (eXtended Markup Langauge). En 2019, le codage texte le plus répandu était JSON (JavaScript Object Noation). Emergence du standard PROTOBUG de `Google`.

Codage textuel

- Choisir un protocole pour séparer les champs, pour encoder des données binaires
- Dans le passé, on utilisait souvent un dérivé de XML (eXtended Markup Langauge). En 2019, le codage texte le plus répandu était JSON (JavaScript Object Noation). Emergence du standard PROTOBUG de `Google`.
- Exemple de codage basés sur des fichiers textes XML :SOAP, XML-RPC

Codage textuel

- Choisir un protocole pour séparer les champs, pour encoder des données binaires
- Dans le passé, on utilisait souvent un dérivé de XML (eXtended Markup Langauge). En 2019, le codage texte le plus répandu était JSON (JavaScript Object Noation). Emergence du standard PROTOBUG de Google.
- Exemple de codage basés sur des fichiers textes XML :SOAP, XML-RPC
- Autre exemple CSV (Comma Separated Values) ;

JSON

Format simple de stockage de données textuelles en utilisant des identifiants de ponctation.

- Les accolades définissent un objet (appelé *membre*.

JSON

Format simple de stockage de données textuelles en utilisant des identifiants de ponctation.

- Les accolades définissent un objet (appelé *membre*.
- Un membre contient des paires (clefs,valeurs). La clef est toujours entre guillemets.

JSON

Format simple de stockage de données textuelles en utilisant des identifiants de ponctation.

- Les accolades définissent un objet (appelé *membre*.
- Un membre contient des paires (clefs,valeurs). La clef est toujours entre guillemets.
- Les crochets annoncent un tableau et les virgules séparent les items du tableau.

JSON

Format simple de stockage de données textuelles en utilisant des identifiants de ponctation.

- Les accolades définissent un objet (appelé *membre*.
- Un membre contient des paires (clefs,valeurs). La clef est toujours entre guillemets.
- Les crochets annoncent un tableau et les virgules séparent les items du tableau.
- L'exemple ci-après est inspiré du livre de V. Barra.

JSON

```
{
"entrées" : ["Melon","Oeufs mayo"],
"plats" : {
"Viandes": ["Boeuf", "Poulet"],
"Poisson": ["Meunière"],
"platsComposees": [
{
"nom": "Boeuf bourguignon",
"ingrédients": []
},
{"nom": "Poisson meunière", "quantités": {}}
],
"viandeEtPoisson": true
},
"nombreInvites": 5, "nomHote": "M. Noyer"
}
```

Sérialisation de structure hiérarchique

Cas des arbres binaires

- On peut donner le résultat sous forme de deux listes : les parcours infixe et préfixes. Mais la place utilisée est le double de celle de l'arbre.

Sérialisation de structure hiérarchique

Cas des arbres binaires

- On peut donner le résultat sous forme de deux listes : les parcours infixe et préfixes. Mais la place utilisée est le double de celle de l'arbre.
- On peut aussi faire un parcours préfixe et indiquer les nœuds **NULL** par un symbole comme -1 (cela rend implicitement l'arbre entier).

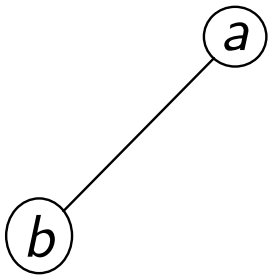


FIGURE – Codage **a b -1 -1 -1**

Sérialisation de structure hiérarchique

Cas des arbres binaires

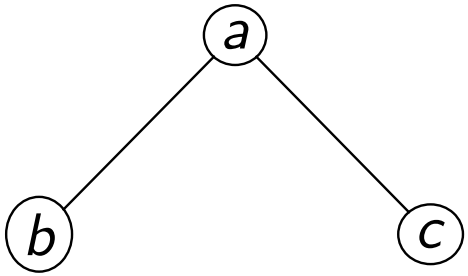


FIGURE – Codage **a b -1 -1 c -1 -1**

Sérialisation de structure hiérarchique

Cas des arbres binaires

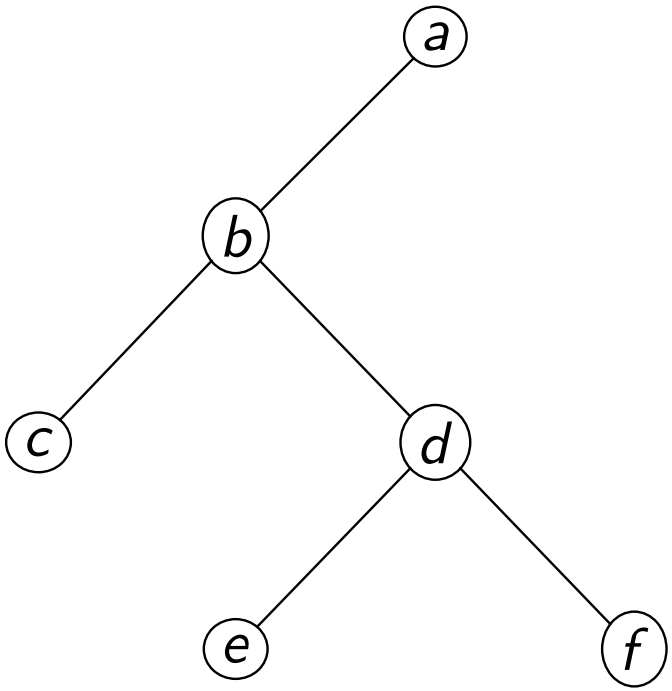


FIGURE – Codage **a b c -1 -1 d e -1 -1 f -1 -1 -1**

(Vincent Barra)

Sérialisation de structure hiérarchique

Cas des arbres *n*-aires

Un symbole (comme $*$) indique la fin de la liste des fils.

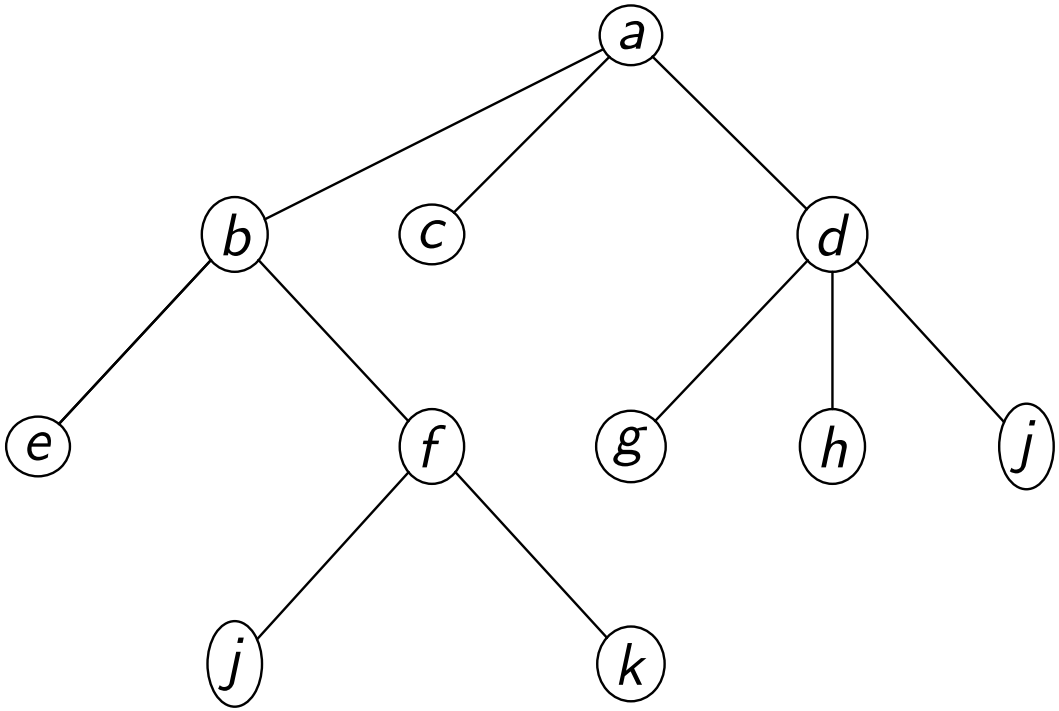
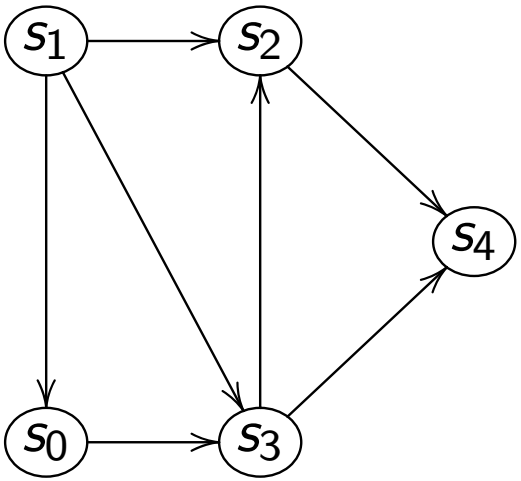


FIGURE – Codage **a b e** $*$ **f j** $*$ k $*$ $*$ $*$ c $*$ d g $*$ h $*$ j$*$ $*$ $*$

(Vincent Barra)

Sérialisation de structure hiérarchique

Cas des graphes orientés



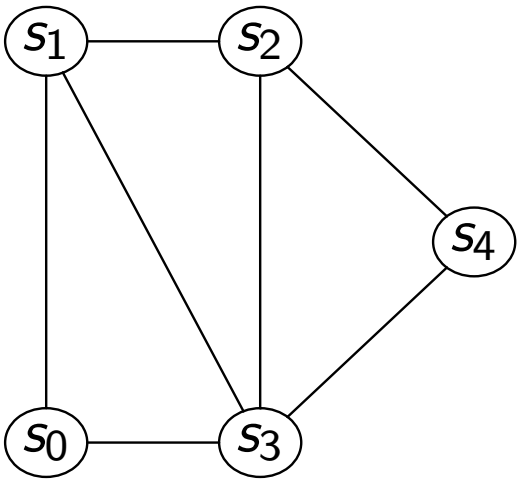
Graphe orienté

```
{
    "0" :  [3] ,
    "1" :  [0 ,2 ,3] ,
    "2"    :  [4] ,
    "3"    :  [2 ,4]
    "4"    :  []
}
```

Sérialisation

Sérialisation de structure hiérarchique

Cas des graphes non orientés



Graphe non orienté

```
{
    "0" :  [1 ,3] ,
    "1" :  [0 ,2 ,3] ,
    "2"    :  [1 ,3 ,4] ,
    "3"    :  [1 ,2 ,4]
    "4"    :  [2 ,3]
}
```

Sérialisation

Rqe : on peut également donner le code caml

[[1 ;3] ;[0 ;2 ;3] ;[1 ;3 ;4] ;[1 ;2 ;4] ;[2 ;3]]

c'est aussi une sérialisation