

COMMENT :

**ARCHIVER ET
RÉFÉRENCER LES
CODES SOURCES
DANS HAL**

2 TYPES DE DÉPÔTS SONT POSSIBLES :

1. DÉPÔT SOURCE :

* LES FICHIERS SOURCES SONT EN LOCAL

1. Préparer le logiciel (localement)

- ajout des fichiers AUTHORS, LICENSE & README
- compresser les documents en .zip /tar.gz
- se connecter au portail [HAL LIRMM](#) et cliquer sur le bouton rouge +Déposer/+Upload
- choisir le type de dépôt logiciel comme indiqué sur la figure ci-dessous

Je sélectionne un type de document Quitter

Choisir le type de document qui correspond à votre publication. L'icône "i" indique que des sous-types de documents sont disponibles et peuvent être sélectionnés.

Article dans une revue	Chapitre d'ouvrage	Pré-publication, Document de travail	Vidéo
Communication dans un congrès	Article de blog scientifique	Rapport	Son
Poster de conférence	Notice d'encyclopédie ou de dictionnaire	Thèse	Carte
Proceedings/Recueil des communications	Traduction	HDR	Logiciel
N° spécial de revue/special issue	Brevet	Cours	
Ouvrages	Autre publication scientifique	Image	

Chargez les métadonnées à partir d'un identifiant.
Les informations associées à cet identifiant permettront de compléter automatiquement votre dépôt.

DOI: 10.xxxx

Récupérer les métadonnées

Auteurs et affiliations

Auteurs

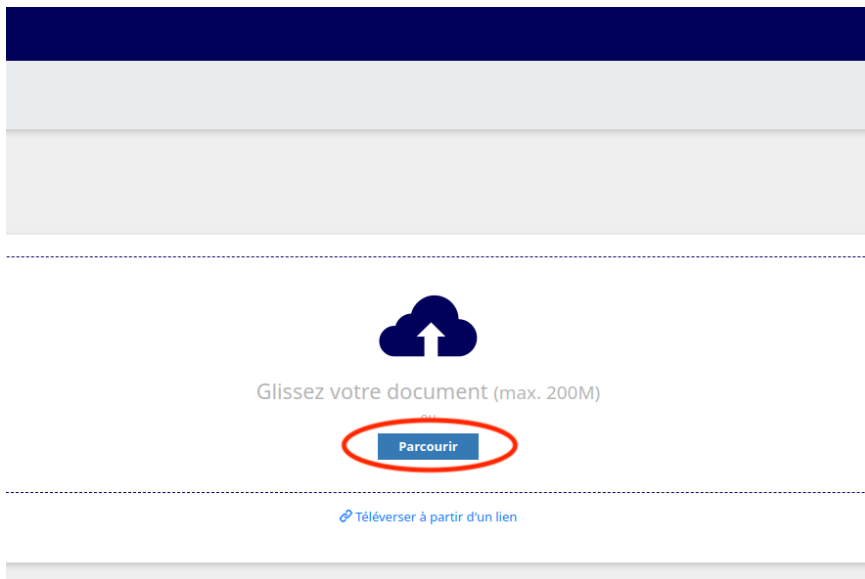
Ajouter un auteur

Ajouter une liste d'auteurs Ajouter les auteurs d'une structure Ajouter mes auteurs Affiler les auteurs Supprimer toutes les affiliations

Informations principales Informations nécessaires pour la citation

Dépôt des fichiers sources compressés
choix du type de document

2. Déposer l'archive compressée (comme sur la figure ci-dessous)



Dépôt des fichiers sources compressés

3. Compléter les métadonnées :

- ajouter des métadonnées générales
- ajouter des métadonnées spécifiques aux logiciels
- renseigner les auteurs
- valider le dépôt

2. DÉPÔT SWHID :



SI LE CODE EST SUR UN CODE REPOSITORY

1. Préparer le logiciel (sur un code repository)

- Ajout des fichiers AUTHORS, LICENSE, README & codemeta.json
- Archiver le logiciel sur Software Heritage : [Save code now](#) (copier l'url du code repository)
- Une fois le code archivé, copier le Software Heritage Identifier (SWHID) dans le champ prévu à cet effet (voir les figures ci-dessus)

2. Compléter les métadonnées

- Ajouter le domaine
- Contrôler les entrées codemeta
- Compléter les métadonnées
- Renseigner les auteurs
- Valider le dépôt

https://codemeta.github.io/codemeta-generator/

CodeMeta generator

ed when generating Codemeta.

Unique identifier <small>(URI, DOI, DOI)</small> <small>such as ISBN, GTIN codes, UUID, etc. https://schema.org/identifier</small>	<input type="text"/>	Code repository <small>https://github.com/youarepothos</small>	<input type="text"/>	Run-time environment Programming Language <small>C++</small>	<input type="text"/>	Current version Version number <small>2022-10-24</small>	<input type="text"/>		
	Application category <small>(Cryptographic primitive)</small>		<input type="text"/>		Continuum integration <small>https://travis-ci.org/youarepothos</small>		Runtime Platform <small>NET, JVM</small>	<input type="text"/>	Release date <small>2022-10-24</small>
	Keywords <small>(Cryptographic primitives, class groups of quadratic imaginary run)</small>		<input type="text"/>		Issue tracker <small>https://github.com/youarepothos/issues</small>		Operating System <small>Android 3.6, Linux, Windows, macOS</small>	<input type="text"/>	Download link <small>https://github.com/youarepothos</small>
	Funding <small>(DOI, DOI, DOI)</small>		<input type="text"/>		Related links <input type="text"/>		Other software requirements <small>Python 3.4 https://github.com/pstf/requests</small>	<input type="text"/>	Release notes <small>Change log Bug fixes</small>
	Funder <small>(University of Pisa)</small>		<input type="text"/>		<input type="text"/>		<input type="text"/>	<input type="text"/>	<input type="text"/>

Authors and contributors can be added below

Author #2 <input type="text"/>	Author #3 <input type="text"/>	Author #4 <input type="text"/>
Given name <input type="text"/>	Given name <input type="text"/>	Given name <input type="text"/>
Family name <input type="text"/>	Family name <input type="text"/>	Family name <input type="text"/>
E-mail address <input type="text"/>	E-mail address <input type="text"/>	E-mail address <input type="text"/>
URL <input type="text"/>	URL <input type="text"/>	URL <input type="text"/>
Affiliation <input type="text"/>	Affiliation <input type="text"/>	Affiliation <input type="text"/>

Outil CodeMeta generator

Enter a SWHID to resolve or keyword(s) to search for in origin URIs

<https://github.com/youarepothos/codemeta-generator>

24 October 2022, 10:16:41 UTC

Code Branches (0) Releases (0) Visits

Branch: HEAD 16c7627

Tip revision: [16c7627](https://github.com/youarepothos/codemeta-generator/commit/16c7627) authored by Cyril Bouvier on 24 October 2022

File	Mode	Size
benches	-rw-r--r--	67 bytes
cmakeutils	-rw-r--r--	4.8 KB
doc	-rw-r--r--	34.2 KB
src	-rw-r--r--	326 bytes
tests	-rw-r--r--	
gitlab-ci.yml	-rw-r--r--	
AUTHORS	-rw-r--r--	
CMakeLists.txt	-rw-r--r--	
LICENSE	-rw-r--r--	
README.md	-rw-r--r--	

README.md

BICYCL

BICYCL Implements Cryptography in Class groups

BICYCL is an Open Source C++ library that implements arithmetic in the ideal class groups of imaginary quadratic fields, together with a set of cryptographic primitives based on class groups.

The documentation is available at <https://crypto.lirmm.net/bicycl>

Permalink: <https://sw.hydroxymol.org/uri/16c7627>

Select below a type of object currently browsed in order to display its associated SWHID and permalink.

- directory
- revision
- snapshot

Add contextual information

Copy identifier Copy permalink

copy SWHID

Product - Solutions - Open Source - Pricing

Search Sign in Sign up

of requests Actions Projects Wiki Security Insights

master 20 branches 42 logs Go to file Code + About

Roberto Di Cosma Update biblatex snippet 669987 on Nov 25, 2022 206 commits

config	Use Array.create_final instead of Obj.dropping support for occant 4.02.3	2 years ago
example	Add support for OCaml 5.0	3 months ago
src	Add support for OCaml 5.0	3 months ago
tests	Only run tests/floatscale on 64bit architectures	2 years ago
ghignore	Version and fix parmap.opam	4 years ago
AUTHORS	Update URL in AUTHORS	4 years ago
CHANGES	Update changelog	2 years ago
LICENSE	Clarified LICENCE and origin of bylaws code.	12 years ago
Makefile	Freshen up the documentation.	4 years ago
README.md	Updated documentation to describe the keeporder parameter.	3 months ago
codemeta.json	Bump version number in codemeta.json	3 months ago
dune-project	Flat float array compiler option must not be used	2 years ago
parmap.bib	Update biblatex snippet	3 months ago
parmap.opam	Revert "Add opam template to gh-testing"	2 years ago

README.md

Parmap in a nutshell

Parmap is a minimalistic library allowing to exploit multicore architecture for OCaml programs with minimal modifications: if you want to use your many cores to accelerate an operation which happens to be a map, fold or mapfold (map-reduce), just use Parmap's `parmap`, `parfold` and `parmapfold` primitives in place of the standard `List.map` and friends, and specify the number of subprocesses to use by the optional parameter `nbcores`.

See the `examples` directory for a couple of running programs.

DO'S and DONT'S

Parmap is not meant to be a replacement for a full fledged implementation of parallelism skeletons (map, reduce, pipe, and the many others described in the scientific literature since the end of the 1980's, much earlier than the specific implementation by Google engineers that popularised them). It is meant, instead, to allow you to quickly leverage the side processing power of your extra cores, when handling some heavy computational load.

About

Parmap is a minimalistic library allowing to exploit multicore architecture for OCaml programs with minimal modifications.

rdicosma.github.io/parmap/

Readme
View license
89 stars
8 watchers

Jan 17, 2020, 4:51 PM GMT+1

Releases (2)

Update for OCaml 5.0 [View](#)
on Jan 2

+ 11 releases

Packages

No packages published

Contributors (8)

+ 7 contributors

Languages


- OCaml 92.6%
- Shell 2.1%
- C 4.8%
- Blaze 0.5%

Fichiers à ajouter au code source

Dépôt SWHID

ogiciel [changer](#)

Fichier(s) Je dépose mes fichiers



Glissez votre document (max. 200M)

[Parcourir](#)

[Téléverser à partir d'un lien](#)

Métadonnées. Je renseigne mon dépôt

Extraction automatique

Chargez les métadonnées à partir d'un identifiant
Les informations associées à cet identifiant permettront de compléter automatiquement votre dépôt.

SWHID

[Récupérer les métadonnées](#)

Auteurs et affiliations

Récupérer les métadonnées à partir du SWHID

Copier le code et l'enregistrer dans un fichier codemeta.json puis le déposer sur le dépôt Git. Ce fichier permettra de pré-remplir les métadonnées dans HAL.

Contributors

Order of contributors does not matter.

codemeta.json

```
{
  "@context": "https://doi.org/10.5063/schema/codemeta-2.0",
  "@type": "SoftwareSourceCode",
  "license": "https://spdx.org/licenses/GPL-3.0-or-later",
  "codeRepository": "https://gite.lirmm.fr/crypto/bicycl",
  "dateCreated": "2021-01-01",
  "datePublished": "2022-01-01",
  "dateModified": "2022-10-24",
  "downloadUrl": "https://gite.lirmm.fr/crypto/bicycl",
  "name": "BICYCL",
  "version": "1.0",
  "description": "BICYCL is an Open Source C++ library that implements arithmetic in the ideal class groups of imaginary quadratic fields,
  "applicationCategory": "Cryptographic primitive",
  "developmentStatus": "active",
  "referencePublication": "https://hal-lirmm.ccsd.cnrs.fr/ECO/lirmm-03863678v1",
  "keywords": [
    "cryptographic primitives",
    "class groups of quadratic imaginary number fields",
    "cryptosystem"
  ],
  "programmingLanguage": [
    "C++"
  ],
  "author": [
    {
      "@type": "Person",
      "givenName": "Cyril",
      "familyName": "Bouvier",
      "email": "cyril.bouvier@lirmm.fr",
      "affiliation": {
        "@type": "Organization",
        "name": "LIRMM"
      }
    },
    {
      "@type": "Person",
      "id": "https://orcid.org/0000-0001-9362-2869",
      "givenName": "Laurent",
      "familyName": "Imbert",
      "email": "laurent.imbert@lirmm.fr",
      "affiliation": {
        "@type": "Organization",
        "name": "ECO, LIRMM"
      }
    },
    {
      "@type": "Person",
      "givenName": "Guilhem",
      "familyName": "Castagnos",
      "affiliation": {
        "@type": "Organization",
        "name": "Institut de Mathématiques de Bordeaux"
      }
    },
    {
      "@type": "Person",
      "id": "https://orcid.org/0000-0001-6464-1139",
      "givenName": "Fabien",
      "familyName": "Laguillaumie",
      "email": "fabien.laguillaumie@lirmm.fr",
      "affiliation": {
        "@type": "Organization",
        "name": "ECO, LIRMM"
      }
    }
  ]
}
```

Le fichier codemeta.json

Le référencement de logiciels dans HAL à partir d'un identifiant SWHID ou de la création d'une notice permet de construire un catalogue et de créer des citations dans plusieurs formats (TEI, BibTeX, CodeMeta...) et d'alimenter les rapports d'activité du chercheur.

and Software Heritage: building a curated software catalog

The screenshot displays the HAL website interface for a software entry. On the left, a diagram illustrates the workflow: a researcher pushes code to a repository (git), which is then mirrored to HAL. The HAL interface shows the entry for 'LibTool' (SWHID: hal-02130801), including its description, version history, and download options. On the right, a code snippet for 'config.m4' is shown, detailing the software's license (GNU Lesser General Public License) and copyright information.

with minimal user overhead

Catalogue de logiciels dans HAL