

# Xcrypt 言語仕様

E-Science Group, Nakashima Laboratory, ACCMS, Kyoto University

平成 23 年 12 月 19 日

## 目次

<b>1</b>	<b>Overview</b>	<b>1</b>
<b>2</b>	<b>用語</b>	<b>2</b>
<b>3</b>	<b>動作環境</b>	<b>2</b>
<b>4</b>	<b>Xcrypt スクリプト</b>	<b>3</b>
4.1	スクリプトの構成 . . . . .	3
4.2	標準 API . . . . .	4
4.2.1	builtin::prepare . . . . .	4
4.2.2	submit . . . . .	8
4.2.3	sync . . . . .	9
4.2.4	spawn . . . . .	9
4.2.5	builtin::join . . . . .	9
<b>5</b>	<b>Xcrypt におけるジョブの管理</b>	<b>9</b>
<b>6</b>	<b>ユーザ環境スクリプト</b>	<b>9</b>
<b>7</b>	<b>バッチスケジューラ定義スクリプト</b>	<b>9</b>
<b>8</b>	<b>コマンドラインインターフェース</b>	<b>9</b>
<b>9</b>	<b>拡張ライブラリ</b>	<b>9</b>

## 1 Overview

In using a high-performance computer, we usually commit job processing to a job scheduler. At this time, we often go through the following procedures:

- to create a script in its writing style depending on the job scheduler,
- to pass the script to the job scheduler, and
- to extract data from its result, create another script from the data, and pass it to the job scheduler.

However, such procedures require manual intervention cost. It therefore seems better to remove manual intervention in mid-processing by using an appropriate script language. Xcrypt is a script language for job parallelization. We can deal with jobs as objects (called *job objects*) in Xcrypt and manipulate the jobs as well as objects in an object-oriented language. Xcrypt provides some functions and modules for facilitating job generation, submission, synchronization, etc. Xcrypt makes it easy to write scripts to process job, and supports users to process jobs easily.

## 2 用語

**バッチスケジューラ** スーパーコンピュータ等大規模計算システムにおける計算資源の利用状況を管理し、ユーザからシステム上で行いたい処理（ジョブ）の要求があれば、資源の空き状況などに基づいて適切にジョブに計算資源を割当て、実行させるソフトウェア。NQS, SGE, Torque など。ユーザは実行したい処理や実行に必要とする計算資源量などを記述したスクリプト（ジョブスクリプト）を入力としてジョブを投入すると、バッチスケジューラはそれを自身が管理する「ジョブキュー」にいったん追加し、ジョブの実行が可能と判断されたタイミングでキューから取り除き、ジョブへの計算資源の割当て・実行指示を行う。

**request ID** バッチスケジューラが、投入されたジョブを識別するためのユニークな ID。

**ジョブオブジェクト** Xcrypt 上において、これから投入しようとする、あるいは投入したジョブを表現するオブジェクト。下記のジョブ ID の文字列などをメンバに持つ。

**ジョブ ID** Xcrypt 上でジョブオブジェクトを識別するための文字列。基本的に request ID と 1 対 1 対応である。<sup>1</sup>

## 3 動作環境

- Bourne shell 互換のシェルスクリプト
- Perl Version 5.8.5 以上。なお、以下の CPAN パッケージを利用する（Xcrypt の配布パッケージに含まれる）。

---

<sup>1</sup> 一度投入したジョブが失敗・ユーザによるキャンセルなどにより再投入された場合には複数の request ID が 1 つのジョブ ID に対応することになるが、その場合でも最新の request ID とジョブ ID との対応は一つになる。

- Sherzod Ruzmetov’s Config-Simple,
  - Marc Lehmann’s Coro (where conftest.c is not contained), EV,
  - Gurusamy Sarathy’s Data-Dumper,
  - Graham Barr’s Error,
  - Joshua Nathaniel Pritikin’s Event,
  - Salvador Fandiño’s Net-OpenSSH,
  - Daniel Muey’s Recursive,
  - H.Merijn Brand and Jochen Wiedmann’s Text-CSV\_XS, and
  - Marc Lehmann’s AnyEvent, common::sense, and Guard.
- バッチスケジューラがインストールされた環境では、そのバッチスケジューラ用の設定ファイルを記述することにより、Xcrypt からそのバッチスケジューラのジョブを投入することができる。バッチスケジューラは、以下の要件をみたす必要がある。
    - ジョブの投入をバッチスケジューラに依頼し、投入されたジョブの request ID を含むメッセージを標準出力に出力するジョブ投入コマンド (qsub 等) を提供すること。
    - ジョブ投入コマンドにより投入され実行待ちとなっているジョブおよび実行中のジョブの request ID の一覧を含むメッセージを標準出力に出力するジョブ確認コマンド (qstat 等) を提供すること。
    - 実行待ちあるいは実行中のジョブの request ID をパラメータとして与えると、そのジョブの実行を取り消すジョブ中止コマンド (qdel 等) を提供すること。
    - ジョブの実行が行われるノードと、ジョブ投入・確認・中止コマンドを実行するノードで (NFS 等により) 共有されるファイルシステムが存在すること。

## 4 Xcrypt スクリプト

### 4.1 スクリプトの構成

Xcrypt スクリプトは以下の構成を持つテキストファイルであり、“*.xcr*” を拡張子とするファイル名で保存されなければならない。

```
use qw([module-name1 ... module-namen] core);
script-body
```

ここで、*module-name*<sub>*k*</sub> ( $1 \leq k \leq n$ ) はこの Xcrypt スクリプトが使用するモジュール名である。  
*script-body* は Xcrypt スクリプト本体であり、追加の use 文を含むほぼ通常の Perl プログラムを書くことができる。ただし、以下の点において Perl とは異なる。

- デフォルトのネームスペース名は main ではなく user である .
- 4.2 節で述べる Xcrypt のコア機能のほか ,  $module-name_1 \dots module-name_n$  のモジュールの機能を利用することができる . また , 3 節で列挙した CPAN モジュールが暗黙のうちに読み込まれている .

## 4.2 標準 API

### 4.2.1 builtin::prepare

#### 引数

- *%template*: ジョブテンプレート

#### 返り値

- スカラコンテキストにおいては生成したジョブオブジェクトの数
- リストコンテキストにおいては生成した全てのジョブオブジェクトを要素に持つ配列

引数として与えられたハッシュオブジェクト ( ジョブテンプレート ) の情報をもとに , 0 個以上のジョブオブジェクトを生成し , 返り値として返す . ジョブテンプレートは Perl のハッシュオブジェクトであり , 以下の名前のメンバを持つことができる . これ以外の名前のメンバが含まれていた場合は , 警告メッセージを表示し , そのメンバは無視される . また , `id` , `id@` いずれかの指定は必須であり , 指定なしに呼び出された場合はエラーを発生する .

1. `RANGEn` (  $n$  は自然数 ) , `RANGES` . ただし , この両者を同時に使用してはならない .
2. `id`
3. `exen` (  $n$  は自然数 )
4. `argn_m` (  $n, m$  は自然数 )
5. `exe`, `initially`, `finally`, `env`, `transfer_variable`, `transfer_reference_level`, `not_transfer_info`, `initially`, `before`, `before_to_job`, `before_return`, `before_bkup`, `before_in_job`, `before_in_xcrypt`, `before_in_xcrypt_return`, `finally`, `after`, `after_to_job`, `after_return`, `after_bkup`, `after_in_job`, `after_in_xcrypt`, `after_in_xcrypt_return`, `cmd_before_exe`, `cmd_after_exe`, `header`
6. JS\_で始まるメンバ名
7. :で始まるメンバ名
8. `buildin::add_key` 関数によって登録されたメンバ名
9. `buildin::add_prefix_of_key` 関数によって登録された文字列から始まるメンバ名

10. 2.-9. のメンバ名の最後に@を追加したメンバ名．ただし，同じメンバ名に対して@を追加した名前と追加していない名前のメンバを同時に指定してはならない．

prepare は呼び出されるとまず，以下の通りジョブオブジェクト（列）を生成し，メンバの設定を行う．

- ジョブテンプレートが  $RANGE_n$  ,  $RANGES$  をメンバに持たない場合  
単一のジョブオブジェクトを生成し，ジョブテンプレートが持つ全てのメンバと同じ名前・値のメンバを設定する．
- ジョブテンプレートが  $RANGE_0, \dots, RANGE_n$  をメンバに持つ場合  
 $RANGE_0, \dots, RANGE_n$  の値は，それぞれ Perl の配列への参照  $@A_0, \dots, @A_n$  でなければならず，ジョブオブジェクトは

$$R = \{(i_0, \dots, i_n) \mid i_0 \leq \#A_0, \dots, i_n \leq \#A_n\} \quad (i_0, \dots, i_n \text{ は自然数})$$

の各要素に対応してそれぞれ生成する． $R$  の要素  $(i_0, \dots, i_n)$  に対応するジョブオブジェクトを  $J(i_0, \dots, i_n)$  とすると，ジョブテンプレートが “name” または “name@” の名前のメンバを  $vname$  を値として持つような  $name$  それぞれに対して（前述の通り，ジョブテンプレートが “name” と “name@” を同時にメンバとして持つことはない．）， $J(i_0, \dots, i_n)$  のメンバを以下のように設定する． $RANGE_0, \dots, RANGE_n$  および  $RANGES$  に対しては以下の設定は行わない．また，以下によるもの以外に，メンバ VALUE が  $\$A[i_0], \dots, \$A[i_n]$  を値とする配列への参照を値として設定される．

1. ジョブテンプレートが “name” をメンバとして持つ場合

(a)  $name = id$  の場合

$J(i_0, \dots, i_n)$  のメンバ  $id$  に

$$vname.sep.i_0 \dots sep.i_n$$

を設定する．ただし， $sep$  は `buidin::set_separator` 関数で設定されたセパレータ文字列である（デフォルトは ‘\_’ ）．

(b)  $name \neq id$  の場合

$J(i_0, \dots, i_n)$  のメンバ “name” に  $vname$  を値として設定する．

2. ジョブテンプレートが “name@” をメンバとして持つ場合

- (a)  $vname$  が配列  $@V$  への参照の場合  $J(i_0, \dots, i_n)$  のメンバ “name” の値として， $\$V[count]$  を設定する．ここで， $count$  は `prepare` 関数の呼び出しで生成される全てのジョブオブジェクトを直列に並べたときの通し番号であり，以下の式で定まる．

$$count = i_n \times B_{n-1} + \dots + i_1 \times B_0 + i_0$$

$$\text{where } B_k = \prod_{j=0}^k (\#A_j + 1)$$

(b)  $vname$  が関数への参照の場合

参照先の関数  $vname$  が以下のように呼び出され，その戻り値を  $J(i_0, \dots, i_n)$  のメンバ “name” の値として設定する．

- ジョブテンプレートへの参照が第 1 引数として渡される
- $\$A[i_0], \dots, \$A[i_n]$  の値がそれぞれ第 2 引数–第  $(n+2)$  引数として渡される
- 関数の実行中、変数`$user::self` を用いて生成途中のジョブオブジェクト  $J(i_0, \dots, i_n)$  への参照にアクセスできる。このジョブオブジェクトには少なくとも上記 1. により決定されるメンバが設定済みである。
- 関数の実行中、変数`@user::VALUE` を用いて 配列  $(\$A[i_0], \dots, \$A[i_n])$  にアクセスできる。

(c)  $vname$  がスカラー値への参照の場合

$J(i_0, \dots, i_n)$  のメンバ” $name$ ” の値として、 $vname$  の参照先のスカラー値を設定する。

(d) それ以外の場合

警告を発生し、 $J(i_0, \dots, i_n)$  のメンバ” $name$ ” の値として任意の値を設定する。

- ジョブテンプレートが RANGES をメンバに持つ場合

RANGES の値は、配列への参照  $R_0, \dots, R_n$  からなる配列への参照

$$[R_0, \dots, R_n]$$

でなければならない、

$$\text{RANGE0} = R_0, \dots, \text{RANGEn} = R_n$$

が設定された場合と同様にジョブオブジェクト列の生成およびメンバの設定を行う。

ジョブオブジェクト（列）の生成とメンバ設定の終了後、各ジョブオブジェクトに対して、クラスメソッド `module-name1::new, \dots, module-namen::new, core::new` のうち定義されている最初のものを呼び出す。new メソッドには第 1 引数としてモジュール名の文字列`$class`、第 2 引数として対応するジョブオブジェクトへの参照`$self` が渡される。`core::new` 以外の各 new メソッドでは、モジュールの機能を達成するためにジョブオブジェクト生成時に必要となる処理が行われる。また、これらのメソッドは、

```
$self = $class->NEXT::new($class, $self);
```

に相当する処理を 1 度だけ実行することにより、次の new メソッドを呼び出す（すなわち、`core::new` が必ず 1 度呼び出される）ことを保証するように実装されている。

なお、`core::new` は Xcrypt システムが提供するものであり、必ず定義されている。`core::new` メソッドは、ジョブオブジェクトに対し、以下の処理を行う。ただし、以下において  $id$  は当該ジョブオブジェクトのメンバ  $id$  の値として設定された文字列、 $sched$  はデフォルトスケジューラ名（??節）である。

- メンバ `workdir` が未設定であれば、値を文字列”.”として設定する。
- メンバ `env` が未設定であれば、値をデフォルトの環境オブジェクト（??節）への参照として設定する。

- メンバ JS\_stdout が未設定であれば、値を文字列"*id\_stdout*"として設定する。
- メンバ JS\_stderr が未設定であれば、値を文字列"*id\_stderr*"として設定する。
- メンバ jobscript\_header, jobscript\_body が未設定であれば、値を空の配列への参照として設定する。
- メンバ jobscript\_file が未設定であれば、値を文字列"*id\_env\_stdout*"として設定する。
- メンバ before\_in\_job\_file が未設定であれば、値を文字列"*id\_before\_in\_job.pl*"として設定する。
- メンバ exe\_in\_job\_file が未設定であれば、値を文字列"*id\_exe\_in\_job.pl*"として設定する。
- メンバ after\_in\_job\_file が未設定であれば、値を文字列"*id\_after\_in\_job.pl*"として設定する。
- メンバ qsub\_options が未設定であれば、値を空の配列への参照として設定する。
- メンバ not\_transfer\_info が未設定であれば、値を 'dumped\_environment', 'before\_in\_job\_script', 'exe\_in\_job\_script', 'after\_in\_job\_script' の 4 つの文字列を要素として持つ配列への参照として設定する。not\_transfer\_info が定義済みであり、かつその値が配列への参照であれば、これら 4 つの文字列をその配列に追加する。それ以外の場合、警告を表示し、メンバの値をこれら 4 つ文字列を持つ配列への参照に上書きする。
- メンバ cmd\_before\_exe, cmd\_after\_exe が未設定であれば、値を空の配列への参照として設定する。
- メンバ header が未設定であれば、値を空の配列への参照として設定する。
- ジョブの状態(??節)を initialized に遷移させ、直後に prepared に遷移させる。<sup>2</sup>

new メソッドの呼び出し完了後、以下の通り値を返す。

- prepare 関数がスカラコンテキストで呼び出されていた場合  
生成したジョブオブジェクトの数
- prepare 関数がリストコンテキストで呼び出されていた場合  
生成した全てのジョブオブジェクトの参照からなる配列。配列中の参照の順序は任意である。

---

<sup>2</sup> 現在の Xcrypt においては、initialized と prepared の 2 つの状態を区別する必要があるが、後方互換性のために残している。

#### 4.2.2 submit

##### 引数

- *jobs*: ジョブオブジェクトへの参照の配列

##### 返り値

- スカラコンテキスト: 生成したジョブスレッドの数
- リストコンテキスト: 生成したジョブスレッドの配列

*jobs* に含まれるそれぞれジョブオブジェクトの参照 (以下, この参照を持つ変数を *\$self* とする) に対して, 以下の処理を行う.

1. *\$self->make\_dumped\_environment* メソッド (??節) を呼び出して, Xcrypt の環境をシリアライズした文字列を保存する.
2. 以下の処理を非同期に実行するスレッド<sup>3</sup> (ジョブスレッド) を生成し, そのスレッドを *\$self->{thread}* に代入する.

(a) ジョブオブジェクトの状態に基づき, 以下の処理を行う.

- *\$self* がシグナル (??節) を受けていない場合  
前回の終了状態 (??節) が *finished* の場合, ジョブの状態を *finished* に強制的に遷移させ, ジョブスレッドを終了する. 前回の終了状態がそれ以外の場合, 何も行わない.
- *\$self* が *abort* シグナルを受けている場合  
シグナルを解除する. 前回の終了状態 (??節) が *finished* の場合, ジョブの状態を *finished* に強制的に遷移させ, ジョブスレッドを終了する. 前回の終了状態がそれ以外の場合, 前回の終了状態を削除する.
- *\$self* が *cancel* シグナルを受けている場合  
シグナルの解除, および前回の終了状態の削除 *jwol* 行う.
- *\$self* が *invalidate* シグナルを受けている場合  
ジョブの状態を *finished* に強制的に遷移させ, ジョブスレッドを終了する.

(b) *module-name<sub>1</sub>::initially, ... module-name<sub>n</sub>::initially, core::initially* のうち, 定義されているものをこの順に全て呼び出す (引数は *\$self* への参照).

(c) *\$self->{before\_in\_xcrypt}* が定義されていれば呼び出す (第1引数として *\$self* への参照, ).

以上の処理の後, *submit* がリストコンテキストで呼び出されていた場合は, 各ジョブオブジェクトのメンバ *thread* の値, すなわち生成したジョブスレッドの配列を返り値として返す. 配列におけるスレッドの順序は, *@jobs* における対応するジョブオブジェクトの順序と等しい. スカラコンテキストで呼び出されていた場合は, 生成したジョブスレッドの数を返す.

<sup>3</sup> ここでいう「スレッド」は CPAN の Coro モジュールにおけるスレッドであり, 非同期に動作するが, 複数のスレッドが同時に進行することはない.



4.2.3 sync

4.2.4 spawn

4.2.5 builtin::join

5 Xcrypt におけるジョブの管理

6 ユーザ環境スクリプト

7 バッチスケジューラ定義スクリプト

8 コマンドラインインターフェース

9 拡張ライブラリ