Xcrypt 言語仕様

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

平成 24 年 1 月 26 日

目 次

1 Overview **2**

2 用語 **3**

3 動作環境 **3**

4 Xcrypt スクリプト **4**

4.1 スクリプトの構成 4

4.2 標準 API 4

4.2.1 builtin::prepare 4

4.2.2 builtin::submit 9

4.2.3 builtin::sync 11

4.2.4 builtin::prepare_submit 12

4.2.5 builtin::submit_sync 12

4.2.6 builtin::prepare_submit_sync 13

4.2.7 builtin::spawn 13

4.2.8 builtin::join 14

4.2.9 builtin::find_job_by_id 14

4.2.10 builtin::add_key 14

4.2.11 builtin::add_prefix_of_key 15

4.2.12 builtin::set_separator 15

4.2.13 builtin::get_separator 15

4.2.14 builtin::add_host 15

4.2.15 builtin::xcr_exist 17

4.2.16 builtin::xcr_qx 17

4.2.17 builtin::xcr_system 17

4.2.18 builtin::xcr_mkdir 18

4.2.19 builtin::xcr_copy 18

4.2.20 builtin::xcr_rename 19

4.2.21 builtin::xcr_symlink 19
4.2.22 builtin::xcr_unlink 20
4.2.23 builtin::get_from 20
4.2.24 builtin::put_into 21

5 Xcrypt におけるジョブの管理 21

6 ユーザ設定ファイル 21

7 バッチスケジューラ定義スクリプト 22

8 Xcrypt モジュール 24
8.1 一般的な Xcrypt モジュール 24
8.2 core モジュール 25
8.2.1 new メソッド 25
8.2.2 start メソッド 26
8.2.3 make_jobscript メソッド 27
8.2.4 make_jobscript_header メソッド 28
8.2.5 make_jobscript_body メソッド 28
8.2.6 make_qsub_options メソッド 30
8.2.7 make_before_in_job_script メソッド 31
8.2.8 make_exe_in_job_script メソッド 32
8.2.9 make_after_in_job_script メソッド 32

9 コマンドラインインターフェース 33

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．[1]

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）．
  - 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

[1] 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 との対応は一対一になる．

  - 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 base qw(`[*module-name*$_1$ ... *module-name*$_n$]` core);`
script-body

ここで，*module-name*$_k$($1 \leq k \leq n$) および `core` はこのスクリプトが使用する Xcrypt モジュール（8 節）の名前である（`core` は使用することがが必須のモジュールである）．

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*$_1$ ... *module-name*$_n$ および `core` の Xcrypt 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

解説　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@いずれかの指定はジョブ ID の設定のために必須であり，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 0 以上の整数），RANGES．ただし，この両者を同時に使用してはならない．
2. id
3. exe$n$（n は 0 以上の整数）
4. arg$n$_m（n, m は 0 以上の整数）
5. exe, 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. builtin::add_key 関数によって登録されたメンバ名
9. builtin::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 RANGEn，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 RANGE0, ..., RANGEn をメンバに持つ場合
  RANGE0, ..., RANGEn の値は，それぞれ Perl の配列 @$A_0, \ldots,$ @A_n への参照でなければならず，ジョブオブジェクトは

$$R = \{(i_0, \ldots, i_n) \mid i_0 \leq \$\#A_0, \ldots, i_n \leq \$\#A_n\} \quad (i_0, \ldots, i_n \text{は 0 以上の整数})$$

  の各要素に対応してそれぞれ生成する．R の要素 $(i_0, \ldots, i_n)$ に対応するジョブオブジェクトを $J(i_0, \ldots, i_n)$ 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 v_{name} 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.），$J(i_0, \ldots, i_n)$ のメンバを以下のように設定する．RANGE0, ..., RANGEn 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[$$i_0$], ..., $A[$$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
     $J(i_0, \ldots, i_n)$ のメンバ id に
     v_{name} . sep . i_0 . . . sep . i_n
     を設定する．ただし，sep は builtin::set_separator 関数で設定されたセパレータ文字列（4.2.12 節）である（デフォルトは “_”）.
     (b) $name \neq$ id の場合
     $J(i_0, \ldots, i_n)$ のメンバ”$name$” に v_{name} 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) v_{name} が配列 @V への参照の場合
     $J(i_0, \ldots, i_n)$ のメンバ”$name$” の値として，$V[$$count$] を設定する．ここで $count$は，今回の 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.

$$count = i_n \times B_{n-1} + \ldots + i_1 \times B_0 + i_0$$
$$\text{where} \quad B_k = \prod_{j=0}^{k} (\$\#A_j + 1)$$

     (b) v_{name} が関数への参照の場合
     参照先の関数 v_{name} が以下のように呼び出され，その返り値を $J(i_0, \ldots, i_n)$ 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[$$i_0$], ..., $A[$$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 $J(i_0, \ldots, i_n)$ 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 ($A[i_0], \ldots,$ $A[i_n]$) にアクセスできる．

(c) v_{name} がスカラ値への参照の場合
$J(i_0, \ldots, i_n)$ のメンバ"$name$" の値として，v_{name} の参照先のスカラ値を設定する．

(d) それ以外の場合
警告を発生し，$J(i_0, \ldots, i_n)$ 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 $R_0, \ldots, R_n$ からなる配列への参照

$$[R_0, \ldots, R_n]$$

でなければならず，

$$\texttt{RANGE0} = R_0, \ldots, \texttt{RANGE}n = R_n$$

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．

- ジョブテンプレートが RANGEn，RANGES をメンバに持つかどうかによらず，ユーザ設定ファイル（6 節）の template セクションに記述されている変数名 N と対応する値 V の組それぞれに対して，N を名前とするメンバが上記の操作によって設定されなかった場合，当該メンバを値を V として追加する．

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-name*$_1$::new, ... *module-name*$_n$::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 Xcrypt 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，Xcrypt 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．また，各 new メソッドの返り値は，“bless $*self*, $*class*” の結果である．

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* はデフォルト環境オブジェクト（4.2.14 節）に対応する sched の値である．

- 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，値をデフォルト環境オブジェクト（4.2.14 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*_*sched*_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．[2]

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．

[2] 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ないが，後方互換性のために残している．

4.2.2 builtin::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：渡されたジョブオブジェクトの数
- リストコンテキスト：渡されたジョブオブジェクトの配列自身

解説 @*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. 以下の処理を非同期に実行するスレッド[3]（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) ジョブの状態に応じて，以下の処理を行う．
   - ジョブがシグナル（**??**節）を受けていない場合
     ジョブの前回の終了状態（**??**節）が finished の場合，ジョブの状態を finished 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．
   - ジョブが abort シグナルを受けている場合
     シグナルを解除する．前回の終了状態が finished の場合，ジョブの状態を finished 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，前回の終了状態の設定を削除する．
   - ジョブが 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設定の削除を行う．
   - ジョブが invalidate シグナルを受けている場合
     ジョブの状態を finished に遷移させ，ジョブスレッドを終了する．

   (b) $*self*->initially, *module-name*$_1$::initially, ... *module-name*$_n$::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
   (c) $*self*->{before_in_xcrypt}が定義されていれば呼び出す．
   (d) ジョブの状態に応じて，以下の処理を行う．
   - ジョブがシグナルを受けていない場合
     ジョブの前回の終了状態が設定されていないか initialized, prepared, aborted のいずれかであれば，*module-name*$_1$::before, ... *module-name*$_n$::before のうち，定義されているものをこの順に全て呼び出す．$*self*->{before_to_job}

[3] 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．

が偽であれば，さらに$*self*->before を呼び出す．ジョブの前回の終了状態が上記以外であれば何も行わない．

- ジョブが abort シグナル，あるいは cancel シグナルを受けている場合
  ジョブの状態を aborted に遷移させる．
- ジョブが invalidate シグナルを受けている場合
  ジョブの状態を finished に遷移させる．

(e) ジョブの状態に応じて，以下の処理を行う．

- ジョブがシグナルを受けていない場合
  ジョブの前回の終了状態が submitted, queued, running, done, finished のいずれかであれば何もしない．前回の終了状態が設定されていないか initialized, prepared, submitted, aborted のいずれかであれば，*module-name*$_1$::start, ... *module-name*$_n$::start, core::start のうち定義されている最初のものを呼び出す．start メソッドの本体では，8.1 節に示す—NEXT—を用いたメソッド呼び出し式により，次の start メソッドを呼び出すことができるが，これを行うことは必須ではない．したがって，core::start が呼び出されることは保証されない．ただし，core::start が呼び出されない場合でも，以下のいずれかが保証される．
  - start メソッドの処理によってジョブの状態が submitted, queued にこの順に遷移させられること．さらに，queued に遷移した後，start メソッドの処理またはこのジョブに対応するジョブスレッド以外のスレッドによって，ジョブの状態が running, done にこの順に有限時間内に遷移させられること．
  - start メソッドの処理またはこのジョブに対応するジョブスレッド以外のスレッドによって，ジョブの状態が aborted に有限時間内に遷移させられること．

  core::start が呼び出された場合，ジョブの情報や環境設定などに基づいて，ジョブスクリプトの生成やバッチスケジューラへのジョブの投入が実行される．（詳細な動作は 8.2.2 を参照.）
- ジョブが abort シグナル，あるいは cancel シグナルを受けている場合
  ジョブの状態を aborted に遷移させる．
- ジョブが invalidate シグナルを受けている場合
  ジョブの状態を finished に遷移させる．

(f) ジョブの状態に応じて，以下の処理を行う．

- ジョブがシグナルを受けていない場合
  ジョブの前回の終了状態が設定されていないか initialized, prepared, submitted, queued, running, aborted のいずれかであれば，ジョブが done, finished, aborted のいずれかになるのを待ち合わせる．前回の終了状態が上記以外であれば何もしない．

- ジョブが abort シグナル，あるいは cancel シグナルを受けている場合
  ジョブの状態を aborted に遷移させる．
- ジョブが invalidate シグナルを受けている場合
  ジョブの状態を finished に遷移させる．

(g) ジョブの状態に応じて，以下の処理を行う．

- ジョブがシグナルを受けていない場合
  ジョブの前回の終了状態が設定されていないか initialized, prepared, submitted, queued, running, done, aborted のいずれかであれば，`$`*self*`->{after_to_job}` の真偽を確認し，偽であれば`$`*self*`->after`, を呼び出す．さらに，*module-name*$_n$`::after`, ... *module-name*$_1$`::after` のうち，定義されているものをこの順に全て呼び出す．ジョブの前回の終了状態が finished であれば何も行わない．
- ジョブが abort シグナル，あるいは cancel シグナルを受けている場合
  ジョブの状態を aborted に遷移させる．
- ジョブが invalidate シグナルを受けている場合
  ジョブの状態を finished に遷移させる．

(h) `$`*self*`->{after_in_xcrypt}`が定義されていれば呼び出す．

(i) `core::finally`, *module-name*$_n$`::finally`, ... *module-name*$_1$`::finally`, `$`*self*`->finally` のうち，定義されているものをこの順に全て呼び出す．

(j) ジョブの状態に応じて，以下の処理を行う．

- ジョブがシグナルを受けていないか，invalidate シグナルを受けている場合
  ジョブの状態を finished に遷移させる．
- ジョブが abort シグナル，あるいは cancel シグナルを受けている場合
  ジョブの状態を aborted に遷移させる．

なお，上記における `initially`, `before_in_xcrypt`, `before`, `after`, `after_in_xcrypt`, `finally` の呼び出し時の引数は以下の通りである．

- 第 1 引数：ジョブオブジェクトへの参照`$`*self*
- 第 2 引数以降：`@{$`*self*`->{VALUE}}`（設定されている場合のみ）

また，`start` の呼び出し時には第 1 引数としてジョブオブジェクトへの参照のみが渡される．

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`@`*jobs*自身を，スカラコンテキストで呼び出されていた場合は`@`*jobs*の要素数を返す．

4.2.3 builtin::sync

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：渡されたジョブオブジェクトの数
- リストコンテキスト：渡されたジョブオブジェクトの配列自身

解説　@*jobs*の要素数が 1 以上の場合，この配列の要素から参照されている全てのジョブオブジェクトのジョブスレッドの終了を待ち合わせる．submit 関数適用前のジョブオブジェクトに対して sync が適用されたときの動作は未定義である．

@*jobs*の要素数が 0，即ち無引数で呼び出された場合は，現在所属する最内の join スコープ（4.2.8 節）において生成された全てのジョブスレッドの終了を待ち合わせる．

sync の返り値は，この関数がリストコンテキストで呼び出されていた場合は@*jobs*自身，スカラコンテキストで呼び出されていた場合は@*jobs*の要素数である．

4.2.4　builtin::prepare_submit

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

解説　builtin::prepare(%*template*) の呼び出し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った後，生成された各ジョブオブジェクトに対して new メソッドおよび builtin::submit 関数の適用を行う．

それぞれのジョブオブジェクトに対する new メソッドおよび builtin::submit 関数の適用の順序は，1 つのジョブオブジェクトに対して new と builtin::submit がこの順序で適用されている限り任意である．

prepare_submit の返り値は，生成されたジョブオブジェクト列への参照の配列（リストコンテキスト）あるいは生成されたジョブオブジェクトの数（スカラコンテキスト）である．

4.2.5　builtin::submit_sync

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：渡されたジョブオブジェクトの数
- リストコンテキスト：渡されたジョブオブジェクトの配列自身

解説　以下の呼び出しと等価である．

```
sync (submit (@jobs))
```

4.2.6　builtin::prepare_submit_sync

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

解説　以下の呼び出しと等価である．

```
sync (prepare_submit (%template))
```

4.2.7　builtin::spawn

書式　builtin::spawn {*code*$_{exe}$}[_initially_{*code*$_{initially}$}]
[_before_in_xcrypt_{*code*$_{before\_in\_xcrypt}$}] [_before_{*code*$_{before}$}] [_after_{*code*$_{after}$}]
[_after_in_xcrypt_{*code*$_{after\_in\_xcrypt}$}] [_finally_{*code*$_{finally}$}] [(%*template*)];

解説　以下の呼び出し文と等価である．

```
prepare_submit(
  id => id,
  exe => sub{code_exe},
  initially => sub{code_initially},
  before_in_xcrypt => sub{code_before_in_xcrypt},
  before => sub{code_before},
  after => sub{code_after},
  after_in_xcrypt=> sub{code_after_in_xcrypt},
  finally => sub{code_finally},
  %template_id);
```

ただし，%*template*$_{\_id}$ は%*template* からメンバ id を除いたハッシュオブジェクトであり，*id* は以下のように定められる文字列である．

- %*template*にメンバ id が含まれていればその値.
- %*template*にメンバ id が含まれていなければ，これまで prepare で生成されたジョブオブジェクトのメンバ id の値および前回の実行履歴（**??**節）に登録されているジョブ ID のいずれの文字列とも重複しない，かつ “spawned” で始まる文字列.

4.2.8 builtin::join

書式 builtin::join {$code_{body}$};

解説 新たな「join スコープ」を構成し，そのスコープ内において $code_{body}$を実行する．

join スコープは動的な生存期間を持つ．すなわち，$code_{body}$の実行開始から実行終了までは，構文上 join ブロック外に記述されたコードの実行もこの join スコープに属しているとみなされる．また，join で構成される全ての join スコープの外側にグローバルな join スコープが存在し，どの非グローバルな join スコープにも属さないコード実行は，このグローバル join スコープに属しているとみなされる．

4.2.9 builtin::find_job_by_id

引数

- $*name*: 文字列

返り値 ジョブオブジェクトへの参照または偽

解説 与えられた文字列$*name*をメンバ id の値として持つ，prepare で生成済みのジョブオブジェクトへの参照を返す．そのようなジョブオブジェクトが存在しない場合は，警告を発生し偽を返す.

4.2.10 builtin::add_key

引数

- @*keys*: 文字列の配列

返り値 未定義

解説 与えられた文字列の配列$*keys*の各要素に対し，この関数の実行以降の prepare 関数（4.2.1 節）の実行においてジョブテンプレートのメンバ名として当該文字列を使用することを可能にする．この関数の返り値は未定義である．

4.2.11 builtin::add_prefix_of_key

引数

- @*prefixes*: 文字列の配列

返り値 未定義

解説 与えられた文字列の配列$*prefixes*の各要素に対し，この関数の実行以降の prepare 関数（4.2.1 節）の実行においてジョブテンプレートのメンバ名として当該文字列から始まる任意の文字列を使用することを可能にする．この関数の返り値は未定義である．

4.2.12 builtin::set_separator

引数

- $*str*: 文字列

返り値 未定義

解説 セパレータ文字列を$*str*に設定する．セパレータ文字列は prepare 関数（4.2.1 節）において，ジョブオブジェクトのメンバ id の値を設定する際に用いられる．セパレータ文字列のデフォルトは “_” である．なお，文字列に含まれてよい文字は

```
!#+,-.@\^_~
```

のいずれかであり，それをみたさないセパレータ文字列が設定されていた場合，prepare 関数の実行時にエラーが発生する．

4.2.13 builtin::get_separator

引数 なし

返り値 セパレータ文字列

解説 現在設定されているセパレータ文字列を返す．

4.2.14 builtin::add_host

引数

- $*envhash*: 環境情報を定義するハッシュオブジェクトへの参照

返り値　生成した環境オブジェクト

解説　与えられたハッシュオブジェクトの情報に基づき「環境オブジェクト」を生成し，返り値として返す．$*envhash*は以下のメンバを持つハッシュオブジェクトへの参照である．

- host: ログイン先のホスト情報を “*user*@*hostname*” の形式で記述した文字列．省略不可能．この host を指定した add_host の呼び出し以降の任意の時点で，以下の OpenSSH のシェルコマンド実行によりプロンプトの発生なしに[4]指定した当該ホストへのログインが可能でなければならない．さらに，上記コマンドによるログイン後，Xcrypt が提供する各コマンドがログインシェルから利用可能なように，当該ホストに Xcrypt がインストールされていなければならない．

  % ssh *user*@*hostname*

- wd: 上記のログイン先のホスト上のディレクトリを絶対パスあるいはログイン時のワーキングディレクトリからの相対パスで指定する．省略可能．省略された場合は，ホストへのログインを行った後そのホストにおける環境変数$HOME を参照しその値に設定する．wd が省略され，環境変数$HOME が設定されていなかった場合はエラーが発生する．

- sched: ログイン先でジョブの投入，確認，削除のために利用するバッチスケジューラ名．省略不可能． 絶対パスあるいはログイン時のワーキングディレクトリからの相対パスで指定する．

- xd: ログイン先で Xcrypt の各コマンドの実行ファイルが存在するディレクトリ．絶対パスあるいはログイン時のワーキングディレクトリからの相対パスで指定する．省略可能．省略された場合は，ホストへのログインを行った後そのホストにおける環境変数$XCRYPT を参照しその値に設定する．xd が省略され，環境変数$XCRYPT が設定されていなかった場合はエラーが発生する．

返り値の環境オブジェクトは prepare 関数（4.2.1 節）に与えるジョブテンプレートのメンバ env の値とすることができ，その結果生成されたジョブオブジェクトに対応するジョブの投入処理は add_host による環境オブジェクト生成時に$*envhash*のメンバで指定したホスト・ワーキングディレクトリ上で実行される（詳細は 8.2.2 節）．

この環境オブジェクトはまた，xcr_exist, xcr_qx, xcr_system, xcr_mkdir, xcr_copy, xcr_rename, xcr_symlink, xcr_unlink, get_from, put_into （4.2.15 節–4.2.24 節）の引数として当該ホスト上でのファイル操作を行うために用いることができる．

なお，Xcrypt の起動時に，デフォルトの環境オブジェクトが 1 つ自動的に定義される．デフォルト環境オブジェクトに対応する host は Xcrypt を起動したユーザ名とホスト名を@で連結した文字列，wd は Xcrypt を起動したときのワーキングディレクトリ，sched は sh，xd は環境変数$XCRYPT（ただし，ユーザ設定ファイル（6 節）において environment セクション

[4] たとえば，当該ホストには公開鍵認証によりログインできるようにし，Xcrypt の実行前に ssh_agent により秘密鍵を登録しておくことでプロンプトが発生しないようにしておく．

のユーザ環境変数 host，wd，sched，xd が設定されていた場合は，上記のメンバの値は対応する名前のユーザ環境変数の設定値となる.）デフォルト環境オブジェクトは prepare 呼び出し時のジョブテンプレートで env が省略された際にジョブオブジェクトの env の値として自動的に設定されるほか，グローバル変数 builtin::env_d の値として参照することができる.

4.2.15 builtin::xcr_exist

引数

- $*env*: add_host 関数で生成した環境オブジェクト
- $*path*: パス名

返り値 ファイルまたはディレクトリが存在すれば真，そうでなければ偽.

解説 指定した$*env*に対応するホスト上で，$*path*で指定したファイルまたはディレクトリが存在するかどうかを判定し，存在すれば真，存在しなければ偽を返す.

$*path*は相対パスで指定する．相対パスの基点は add_host 関数による$*env*の生成時に引数のハッシュオブジェクトのメンバ wd で指定したディレクトリである.

4.2.16 builtin::xcr_qx

引数

- $*env*: add_host 関数で生成した環境オブジェクト
- $*command*: コマンド文字列
- $*dir*: ディレクトリ名

返り値 コマンド実行による標準出力の文字列を改行コードで区切った配列

解説 指定した$*env*に対応するホスト上で，$*dir*で指定したディレクトリにワーキングディレクトリを移した後，$*command*をシェルコマンドとして実行し，その標準出力を返す.

$*dir*は相対パスで指定する．相対パスの基点は add_host 関数による$*env*の生成時に引数のハッシュオブジェクトのメンバ wd で指定したディレクトリである.

4.2.17 builtin::xcr_system

引数

- $*env*: add_host 関数で生成した環境オブジェクト
- $*command*: コマンド文字列
- $*dir*: ディレクトリ名

返り値　終了コード

解説　指定した$*env*に対応するホスト上で，$*dir*で指定したディレクトリにワーキングディレクトリを移した後，$*command*をシェルコマンドとして実行し，その終了コードを返す．
　$*dir*は相対パスで指定する．相対パスの基点は add_host 関数による$*env*の生成時に引数のハッシュオブジェクトのメンバ wd で指定したディレクトリである．

4.2.18　builtin::xcr_mkdir

引数

- $*env*: add_host 関数で生成した環境オブジェクト
- $*dir*: ディレクトリ名

返り値　ディレクトリの生成が成功した場合は真，そうでなければ偽．

解説　指定した$*env*に対応するホスト上で，$*dir*ディレクトリが存在するかを確認し，存在しなければディレクトリを作成する．
　$*dir*は相対パスで指定する．相対パスの基点は add_host 関数による$*env*の生成時に引数のハッシュオブジェクトのメンバ wd で指定したディレクトリである．

4.2.19　builtin::xcr_copy

引数

- $*env*: add_host 関数で生成した環境オブジェクト
- $$path_1$$: コピー元のパス名
- $$path_2$$: コピー先のパス名

返り値　コピーに成功した場合は真，そうでなければ偽．

解説　指定した$*env*に対応するホスト上で，$$path_1$$ から$$path_2$$ へのファイルあるいはディレクトリのコピーを実行する．$$path_1$$ がディレクトリであった場合，そのディレクトリ以下の全てのサブディレクトリおよびファイルを全て再帰的にコピーする．また，$$path_2$$ が既存のファイルであった場合にはそのファイルへの上書きコピーが行われ，既存のディレクトリであった場合はそのディレクトリ上に新たなファイル・ディレクトリがコピーとして作成される．$$path_2$$ が既存ディレクトリ上の存在しないファイル名であった場合は，そのディレクトリ上に，$$path_1$$ のファイル名あるいはディレクトリ名をその$$path_2$$ のファイル名に変更した上で新たなファイル・ディレクトリがコピーとして作成される．$$path_2$$ が存在しないディレクトリ上のファイル名であった場合は何も行わない．

$$path_1$, $$path_2$ は相対パスで指定する．相対パスの基点は add_host 関数による$$env$の生成時に引数のハッシュオブジェクトのメンバ wd で指定したディレクトリである．

4.2.20 builtin::xcr_rename

引数

- $$env$: add_host 関数で生成した環境オブジェクト
- $$path_1$: 移動元のパス名
- $$path_2$: 移動先のパス名

返り値 移動が成功した場合は真，そうでなければ偽．

指定した$$env$に対応するホスト上で，$$path_1$ から$$path_2$ へのファイルあるいはディレクトリの移動を行う．$$path_2$ が既存のファイルであった場合にはそのファイルを上書きして移動が行われ，既存のディレクトリであった場合はそのディレクトリ上に新たなファイル・ディレクトリが移動先ファイル・ディレクトリとして作成される．$$path_2$ が既存ディレクトリ上の存在しないファイル名であった場合は，そのディレクトリ上に，$$path_1$ のファイル名あるいはディレクトリ名をその$$path_2$ のファイル名に変更した上で新たなファイル・ディレクトリが移動先ファイル・ディレクトリとして作成される．$$path_2$ が存在しないディレクトリ上のファイル名であった場合は何も行わない．

$$path_1$, $$path_2$ は相対パスで指定する．相対パスの基点は add_host 関数による$$env$の生成時に引数のハッシュオブジェクトのメンバ wd で指定したディレクトリである．

4.2.21 builtin::xcr_symlink

引数

- $$env$: add_host 関数で生成した環境オブジェクト
- $$dir$: ディレクトリ名
- $$target$: リンク先のパス名
- $$link$: シンボリックリンクのパス名

返り値 シンボリックリンクの作成が成功した場合は真，そうでなければ偽．

解説　指定した$*env*に対応するホスト上で，パス名$*target*のシンボリックリンクを$*dir*/$*link*に作成する．$*dir*/$*link*が既存のファイルであった場合には何も行わない．$*dir*/$*link*が既存のディレクトリであった場合はそのディレクトリ上にパス名$*target*からディレクトリ名を取り除いた名前と同名の新たなシンボリックリンクが作成される．$*dir*/$*link* が既存ディレクトリ上の存在しないファイル名であった場合は，そのディレクトリ上に$*link*からディレクトリ名を除いた名前のシンボリックリンクを作成する．$*dir*/$*link*が存在しないディレクトリ上のファイル名であった場合は何も行わない．

$*dir*および$*link*は相対パスで指定する．$*dir*の基点は add_host 関数による$*env*の生成時に引数のハッシュオブジェクトのメンバ wd で指定したディレクトリである．

4.2.22　builtin::xcr_unlink

引数

- $*env*: add_host 関数で生成した環境オブジェクト
- $*path*: パス名

返り値　削除が成功した場合は真，そうでなければ偽．

解説　指定した$*env*に対応するホスト上で，$*path*で指定したファイルあるいはディレクトリを削除する．$*path*がディレクトリであった場合はそのディレクトリ以下のサブディレクトリおよびファイルを全て削除する．$*path*が存在しないファイルであった場合は何も行わない．

$*path*は相対パスで指定する．相対パスの基点は add_host 関数による$*env*の生成時に引数のハッシュオブジェクトのメンバ wd で指定したディレクトリである．

4.2.23　builtin::get_from

引数

- $*env*: add_host 関数で生成した環境オブジェクト
- $*path*: コピー元のパス名
- $*to*（省略可能）: コピー先のディレクトリ．省略した場合は'.' を指定したとみなされる．

返り値　コピーが成功した場合は真，そうでなければ偽．

解説　指定した$*env*に対応するホスト上の$*path*で指定したファイルあるいはディレクトリをXcrypt を実行したホスト上の$*to* にコピーする．$*path*がディレクトリであった場合，そのディレクトリ以下の全てのサブディレクトリおよびファイルを全て再帰的にコピーする．また，$*to*は既存のディレクトリ名でなければならず，そのディレクトリ上に新たなファイル・ディレクトリがコピーとして作成される．$*to*が既存のファイル，あるいは存在しないファイル・ディレクトリであった場合は何も行わない．

$*path*は相対パスで指定する．相対パスの基点は `add_host` 関数による$*env*の生成時に引数のハッシュオブジェクトのメンバ `wd` で指定したディレクトリである．$*to*は相対パスまたは絶対パスで指定する．相対パスの場合，その基点はXcrypt を実行したワーキングディレクトリである．

4.2.24　builtin::put_into

引数

- $*env*: `add_host` 関数で生成した環境オブジェクト
- $*path*: コピー元のパス名
- $*to*（省略可能）: コピー先のディレクトリ．省略した場合は'.' を指定したとみなされる．

返り値　コピーが成功した場合は真，そうでなければ偽．

解説　Xcrypt を実行したホスト上にある$*path*で指定したファイルあるいはディレクトリを，$*env*に対応するホスト上の$*to*にコピーする．$*path*がディレクトリであった場合，そのディレクトリ以下の全てのサブディレクトリおよびファイルを全て再帰的にコピーする．また，$*to*は既存のディレクトリ名でなければならず，そのディレクトリ上に新たなファイル・ディレクトリがコピーとして作成される．$*to*が既存のファイル，あるいは存在しないファイル・ディレクトリであった場合は何も行わない．

$*to*は相対パスで指定する．相対パスの基点は `add_host` 関数による$*env*の生成時に引数のハッシュオブジェクトのメンバ `wd` で指定したディレクトリである．$*path*は相対パスまたは絶対パスで指定する．相対パスの場合，その基点はXcrypt を実行したワーキングディレクトリである．

5　Xcrypt におけるジョブの管理

6　ユーザ設定ファイル

ユーザ設定ファイルは以下の構成を持つテキストファイルである．

[$sectionname_0$]
var_0^0 = val_0^0
...
$var_0^{m_0}$ = $val_0^{m_0}$

...

[$sectionname_n$]
var_n^0 = val_n^0
...
$var_n^{m_n}$ = $val_n^{m_n}$

Xcrypt の起動時，以下の順でファイルの存在を確認し，存在すればそのファイルがユーザ設定ファイルとして読み込まれる．

1. Xcrypt のコマンドラインオプションに`--config` *configfile*が指定されていれば，*configfile*．
2. `$HOME/.xcryptrc`
3. `$XCRYPT/etc/xcryptrc`

読み込まれる設定ファイルに，var_s^i = val_0^0 の記述が存在したとき，セクション $sectionname_s$ のユーザ環境変数 var_s^i が val_0^0 に設定されるという．

`template` セクションのユーザ環境変数 *var*が *val* に設定されると，`prepare` 関数（4.2.1 節）の呼び出しにおいて引数のジョブテンプレートにメンバ *var*が存在しないとき，このメンバが *val*を値として自動的に追加される．

`environment` セクションのユーザ環境変数の設定値は，デフォルト環境オブジェクトの生成時に用いられる（詳細は 4.2.14 節を参照）．

7　バッチスケジューラ定義スクリプト

バッチスケジューラ定義スクリプトは、*schedname*`.pm` という名前でディレクトリ`$XCRYPT/lib/config/` に保存された Perl モジュールスクリプトである．ここで，*schedname* はスケジューラ名を表す任意の名前である．バッチスケジューラ定義スクリプトは上記のディレクトリに複数存在してもよく，Xcrypt 起動時に存在する定義スクリプトが全て読み込まれる（読み込まれる順序は未定義である）．

各バッチスケジューラ定義スクリプトには任意の Perl モジュールを記述することができる．ただし，ハッシュオブジェクトとして定義済のグローバル変数`$jsconfig::jobsched_config` のメンバ *jobsched* に値をハッシュオブジェクトへの参照として設定することが要請される．このハッシュオブジェクトは少なくとも以下のメンバを持たなければならない．

- qsub_command: 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を実行するためのコマンドライン文字列の先頭部分.
- qstat_command: 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を実行するためのコマンドライン文字列の先頭部分.
- qdel_command: 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あるいは実行中のジョブ実行を取り消すジョブ中止コマンドを実行するためのコマンドライン文字列の先頭部分.
- extract_req_id_from_qsub_output: 上記ジョブ投入コマンドを実行した際の標準出力を改行文字で分割した文字列の配列を引数として受け取り，投入されたジョブの request ID を返す関数.
- extract_req_ids_from_qstat_output: 上記ジョブ確認コマンドを実行した際の標準出力を改行文字で分割した文字列のリストを引数として受け取り，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あるいは実行中の全てのジョブの request ID を配列として返す関数.

また，以下の名前のメンバは特別な意味を持つ.

- jobscript_preamble: ジョブ投入のために生成されるジョブスクリプトにおいて先頭行に追加される文字列.
- jobscript_option_*optname*: *optname* は任意の文字列．ジョブオブジェクトのメンバ JS_*optname* の値に対応して，上記ジョブスクリプトにオプション設定のための記述を追加するために用いられる値.
- jobscript_other_options: ジョブ投入のために生成されるジョブスクリプトにおいて，jobscript_option_*optname* によって追加された行に続いて追加される文字列.
- jobscript_workdir: 上記ジョブスクリプトにおいて，プログラムを実行するワーキングディレクトリを参照するために用いられる値.
- jobscript_body_preamble: 上記ジョブスクリプトにおいて，プログラムの実行直前に実行するコマンドを追加するために用いられる値.
- qsub_option_*optname*: *optname* は任意の文字列．ジョブオブジェクトのメンバ JS_*optname* の値に対応して，ジョブ投入コマンド実行時のコマンドラインオプションを追加するために用いられる値.
- qstat_option_*optname*: *optname* は任意の文字列．ジョブオブジェクトのメンバ JS_*optname* の値に対応して，ジョブ確認コマンド実行時のコマンドラインオプションを追加するために用いられる値.

- `qdel_option_`*optname*: *optname* は任意の文字列．ジョブオブジェクトのメンバ `JS_`*optname* の値に対応して，ジョブ中止コマンド実行時のコマンドラインオプションを追加するために用いられる値．

さらに，上記以外の名前のメンバを含んでいてもよい．

これらのメンバの値がどのように利用されるかの詳細は，8.2.2 節を参照のこと．

8 Xcrypt モジュール

8.1 一般的な Xcrypt モジュール

Xcrypt モジュール *module-name* は、*module-name*`.pm` という名前の Perl モジュールスクリプト（Xcrypt モジュールスクリプト）で定義し，ディレクトリ`$XCRYPT/lib/`に保存する．Xcrypt モジュールスクリプトは上記のディレクトリに複数存在してもよい．Xcrypt 起動時，Xcrypt スクリプトの先頭の `use` 文（4.1 節）で指定された Xcrypt モジュールが全て読み込まれる．

Xcrypt モジュールスクリプトは，以下の構成を持つ．

```
package module-name;
module-body
1;
```

module-body は Xcrypt モジュールスクリプトの本体であり，任意の Perl プログラムを書くことができる．これにより，Xcrypt スクリプトから利用するための *module-name* パッケージの変数や関数を定義する．

module-body のトップレベルで `our` を用いて宣言された変数 *module-name*`::`*var* は，Xcrypt スクリプトから同じ名前で参照することができる．

module-body のトップレベルで `sub` を用いて定義された関数（メソッド）*module-name*`::`*func* は Xcrypt スクリプトから同じ関数名で参照できるほか，インスタンスメソッド呼び出し式

$$\$job\texttt{->}func\texttt{(}args\texttt{)}$$

を使って呼び出すことができる．ここで，`$`*job* はジョブオブジェクトへの参照，*args* は *func* に渡す 0 個以上の引数リストである．インスタンスメソッド呼び出し式によって *func* が呼び出された際に渡される引数は，第 1 引数が`$`*job*，第 2 引数以降が *args* である．

Xcrypt スクリプトが

$$\texttt{use base qw(}module\text{-}name_1\ \ldots\ module\text{-}name_n\ \texttt{core);}$$

のように複数のモジュールの使用を宣言しており，かつ *module-name*$_1$`::`*func*, ..., *module-name*$_n$`::`*func*, `core::`*func* のうち複数の定義が存在した場合，これらの *func* は `use` で指定されたモジュール名リストの最も左にあるモジュール名に対応するものから順に高い優先度

を持ち，最も高い優先度を持つ *func* がインスタンスメソッド呼び出し式によって呼び出される．さらに，インスタンスメソッド呼び出し式によって呼び出された関数の本体では，

$*self*->NEXT::*func*(*args*);

という NEXT を用いたメソッド呼び出し式により，次に優先度が高いモジュール名に対応する *func* を呼び出すことができる（そのような *func* が存在しない場合は，メソッドの呼び出しは行われない）．ここで，$*self* は第 1 引数として渡されたジョブオブジェクトへの参照であり次の *func* への第 1 引数として渡され，また，*args* は任意の 0 個以上の引数リストであり次の *func* に第 2 引数以降として渡される．

func として以下の名前を持つ関数（メソッド）は，以下の通り Xcrypt の標準 API 関数や core モジュール（8.2 節）が定義するメソッドから呼び出されるなど特別な意味を持つ．

- new: builtin::prepare 関数から，生成されたジョブオブジェクトを初期化するためにさ呼び出される．詳細は 4.2.1 節を参照．
- initially, before_in_xcrypt, before, start, after, after_in_xcrypt, finally: builtin::submit 関数が生成するジョブスレッドから呼び出される．詳細は 4.2.2 節を参照．

このほか，8.2 節で列挙する core モジュールのメソッドは，各々の仕様をみたすように定義されているため，これらと同名のメソッドを core 以外のモジュールで定義する場合には，そのモジュールにおいて全体の動作を保証しなければならない．さらに，Xcrypt の個別の実装において，それ以外の名前の変数・メソッドが定義される場合がある．これらの名前の変数・メソッドは，実装のドキュメント等において特に明記されない限り，core 以外のモジュールで定義してはならない．

8.2 core モジュール

Xcrypt モジュールのうち，名前が core であるモジュールは Xcrypt システム自体が提供する特別なモジュールであり，以下のメソッドを提供する．

以下の説明において，*CFG*(*param*) は，環境オブジェクト$*self*->{env}に対応するバッチスケジューラの定義スクリプト（7 節）におけるハッシュオブジェクトのメンバ *param* を指す．

8.2.1 new メソッド

引数

- $*class*: モジュール名（='core'）
- $*self*: 初期化対象のジョブオブジェクトへの参照

返り値 “bless $*self* , $*class*” の結果

解説　与えられたジョブオブジェクトのメンバを初期化し，Perl におけるクラスオブジェクトとして返す．メンバの初期化等の詳細は 4.2.1 節を参照．

8.2.2　start メソッド

引数

- $*self*: ジョブオブジェクトへの参照

返り値　特に規定しない

解説　以下の手順で，$*self* に対応するジョブをバッチスケジューラのジョブ投入コマンドを用いて投入する．

1. ジョブオブジェクトの `make_jobscript` メソッド（8.2.3 節）をインスタンスメソッドとして呼び出し，ジョブスクリプトの内容を生成する．
2. ジョブオブジェクトの `make_qsub_options` メソッド（8.2.6 節）をインスタンスメソッドとして呼び出し，ジョブ投入コマンドに付加するコマンドラインオプション文字列を生成する．
3. ジョブオブジェクトの `make_before_in_job_script` メソッド（8.2.7 節）をインスタンスメソッドとして呼び出し，ジョブスクリプトから実行される Perl スクリプトの内容を生成する．
4. ジョブオブジェクトの `make_exe_in_job_script` メソッド（8.2.8 節）をインスタンスメソッドとして呼び出し，ジョブスクリプトから実行される Perl スクリプトの内容を生成する．
5. ジョブオブジェクトの `make_after_in_job_script` メソッド（8.2.9 節）をインスタンスメソッドとして呼び出し，ジョブスクリプトから実行される Perl スクリプトの内容を生成する．
6. 以下の通りファイルを生成し，上記 1. および 3.–5. で生成した文字列をそれぞれ書き込む．
    - $*self*`->{jobscript_header}`と$*self*`->{jobscript_body}`の値となっている参照が指す文字列配列を連結し，その全要素を改行文字で連結した文字列を$*self*`->{jobscript_file}`のパス名で指定されるファイルに書き込む．
    - $*self*`->{before_in_job_script}`の値となっている参照が指す文字列配列を改行文字で連結した文字列を$*self*`->{before_in_job_file}`のパス名で指定されるファイルに書き込む．

- $*self*->{exe_in_job_script}の値となっている参照が指す文字列配列を改行文字で連結した文字列を$*self*->{exe_in_job_file}のパス名で指定されるファイルに書き込む.
- $*self*->{after_in_job_script}の値となっている参照が指す文字列配列を改行文字で連結した文字列を$*self*->{after_in_job_file}のパス名で指定されるファイルに書き込む.

7. 6. で生成した各ファイル *file* に対し，put_into 関数（4.2.24 節）を，第 1 引数に環境オブジェクト$*self*->{env}，第 2 引数に$*self*->{workdir}と *file* を連結したパス名を与えて呼び出すことで，ファイルを転送する．ただし，転送元と転送先が同一である場合は，何も行わない.
8. ジョブの状態を submitted に遷移させる.
9. xcr_qx 関数（4.2.16 節）を，第 1 引数に環境オブジェクト$*self*->{env}，第 2 引数に文字列'*command options scriptfile*'，第 3 引数に$*self*->{workdir}の値となっているワーキングディレクトリを与えて呼び出すことで，$*self*->{env}のホスト上でジョブ投入コマンドを実行し，その標準出力を獲得する．ただし，*command* は *CFG*(qsub_command) の値として定義されている文字列，*options* は$*self*->{qsub_options}の参照先の配列の各要素を空白文字で連結した文字列，*scriptfile*は$*self*->{jobscript_file}の値となっている文字列である.
10. $*self*->{env}に対応するバッチスケジューラの定義スクリプトにおいて extract_req_id_from_qsub_output の値として定義されている関数を，9. で実行した xcr_qx の返り値（文字列のリスト）を引数として呼び出して投入したジョブの request ID を獲得し，その ID を$*self*->{request_id}の値として保存する.

8.2.3 make_jobscript メソッド

引数

- $*self*: ジョブオブジェクトへの参照

返り値 特に規定しない

解説 start メソッド（8.2.2 節）の実行中に呼び出され，ジョブ投入コマンド実行時に必要なジョブスクリプトの内容となる文字列を生成する．このメソッド自体は，make_jobscript_header メソッド（8.2.4 節）および make_jobscript_body メソッド（8.2.5 節）をこの順で呼び出すだけの処理を行い，具体的な生成処理はこれらのメソッドで行われる.

8.2.4 make_jobscript_header メソッド

引数

- $*self*: ジョブオブジェクトへの参照

返り値 特に規定しない

解説 make_jobscript メソッドの実行中に呼び出され，ジョブスクリプトの内容となる文字列のうちヘッダ部分を生成し，その結果を改行文字で区切った文字列配列への参照として $*self*->{jobscript_header}に格納する．

文字列の生成は以下のように行う．

1. *CFG*(jobscript_preamble) が関数であれば，その関数を$*self*を引数として呼び出した結果，文字列であればその文字列に改行文字を加えたもの，配列への参照であればそれらの要素を改行文字で連結し末尾にも改行文字を加えたものをスクリプトに追加する．
2. *CFG*(jobscript_option_*optname*) の値が存在する全ての *optname*に対して以下を行う．
   - *CFG*(jobscript_option_*optname*) が関数の場合，その関数を第 1 引数として $*self*，第 2 引数として文字列'JS_*optname*' を与えて呼び出し，その返り値である文字列の末尾に改行文字を加えたものをスクリプトに追加する．
   - *CFG*(jobscript_option_*optname*) が文字列の場合，$*self*->{JS_*optname*}の値が定義されていれば，この 2 つの文字列を連結させたものに改行文字を加えたものをスクリプトに追加する．$*self*->{JS_*optname*}の値が定義されていなければ何も行わない．
   - *CFG*(jobscript_option_*optname*) が関数でも文字列でもない場合，警告を発生させる．
3. *CFG*(jobscript_other_options) に対して，2. と同様の手続きでスクリプトに文字列を追加する．
4. $*self*->{JS_user_preamble}が定義されている場合，その値が文字列であればそれに改行文字を追加したもの，配列への参照であればそれらの要素を改行文字で連結し末尾にも改行文字を加えたものをスクリプトに追加する．
5. 各実装に応じて，必要なスクリプトを追加する．

8.2.5 make_jobscript_body メソッド

引数

- $*self*: ジョブオブジェクトへの参照

返り値　特に規定しない

解説　make_jobscript メソッドの実行中に呼び出され，ジョブスクリプトの内容となる文字列のうち本体部分を生成し，その結果を改行文字で区切った文字列配列への参照として $*self*->{jobscript_body}に格納する．

文字列の生成は以下のように行う．

1. 文字列'cd *wkdir*' に改行文字を加えた文字列をスクリプトに追加する．ただし *wkdir* は以下のように決定される文字列である．
   - *CFG*(jobscript_other_options) の値が文字列の場合，その値．
   - *CFG*(jobscript_other_options) の値が関数の場合，その関数を$*self*を引数として呼び出した返り値の文字列．
   - *CFG*(jobscript_other_options) の値が未定義の場合，環境オブジェクト$*self*->{env}に対応するワーキングディレクトリと$*self*->{workdir}を連結したパス名．なお，*CFG*(jobscript_other_options) の値が文字列でも関数でもない場合は，警告を発生させたうえで，未定義の場合として扱う．
2. *CFG*(jobscript_body_preamble) が関数であれば，その関数を$*self*を引数として呼び出した結果，文字列であればその文字列に改行文字を加えたもの，配列への参照であればそれらの要素を改行文字で連結し末尾にも改行文字を加えたものをスクリプトに追加する．
3. このジョブスクリプトを実行するプロセスから Xcrypt 本体に，$*self*に対応するジョブの状態を running に遷移させることを促すメッセージを送るコマンドライン文字列（末尾の改行文字を含む）をスクリプトに追加する．
4. $*self*->{cmd_before_exe}の値が配列への参照として定義されていれば，その各要素を改行文字で連結した文字列をスクリプトに追加する．このメンバが配列へ参照以外の値に設定されていた場合は，エラーを発生する．
5. $*self*->{before_in_job_file}のファイルを Perl プログラムとして実行するコマンドライン文字列（末尾の改行文字を含む）をスクリプトに追加する．
6. $*self*->{exe}が定義されている場合
   (a) $*self*->{exe*n*}（*n* は 0 以上の整数）も定義されていた場合，警告を発生する．
   (b) $*self*->{exe}の値が関数であれば，$*self*->{exe_in_job_file}のファイルを Perl をプログラムとして実行するコマンドライン文字列（末尾の改行文字を含む）をスクリプトに追加する．$*self*->{exe}の値が関数でなければ警告を発生し，何も行わない．
7. $*self*->{exe}が定義されていない場合，値が定義されている$*self*->{exe*n*}（*n* は 0 以上の整数）に対し，*n* の値が小さいものから順に以下を行う．

- $*self*->{exe*n*}の値の文字列に$*self*->{exe*n*_*m*}（m は 0 以上の整数）のうち値が定義されているものを m の値が小さいものから順に空白文字を挟みつつ連結し，最後に末尾に改行文字を追加したものをスクリプトに追加する

8. $*self*->{after_in_job_file}のファイルを Perl プログラムとして実行するコマンドライン文字列（末尾の改行文字を含む）をスクリプトに追加する．
9. $*self*->{cmd_after_exe}の値が配列への参照として定義されていれば，その各要素を改行文字で連結した文字列をスクリプトに追加する．このメンバが配列へ参照以外の値に設定されていた場合は，エラーを発生する．
10. このジョブスクリプトを実行するプロセスから Xcrypt 本体に，$*self* に対応するジョブの状態を done に遷移させることを促すメッセージを送るコマンドライン文字列（末尾の改行文字を含む）をスクリプトに追加する．

8.2.6 make_qsub_options メソッド

引数

- $*self*: ジョブオブジェクトへの参照

返り値 特に規定しない

解説 start メソッド（8.2.2 節）の実行中に呼び出され，ジョブ投入コマンド実行時に指定するコマンドラインオプションの内容を生成する．結果は，文字列配列への参照として$*self*->{qsub_options}に格納する．

上記の文字列配列の生成は，空の配列から始めて，以下の手順で末尾に要素を追加していくことで行う．

1. *CFG*(qsub_option_*optname*) の値が存在する全ての *optname* に対して以下を行う．
   - *CFG*(qsub_option_*optname*) が関数の場合，その関数を第 1 引数として$*self*，第 2 引数として文字列'JS_*optname*' を与えて呼び出し，その返り値である配列を末尾に連結する．
   - *CFG*(jobscript_option_*optname*) が文字列の場合，$*self*->{JS_*optname*}の値が定義されていれば，この 2 つの文字列をこの順に配列の末尾に追加する．$*self*->{JS_*optname*}の値が定義されていなければ何も行わない．
   - *CFG*(jobscript_option_*optname*) が関数でも文字列でもない場合，警告を発生させる．
2. $*self*->{JS_user_qsub_options}が定義されている場合，その値が文字列であればそれをこれまでの配列の末尾に追加，配列への参照であればその配列をこれまでの配列の末尾に連結する．

8.2.7 make_before_in_job_script メソッド

引数

- $*self*: ジョブオブジェクトへの参照

返り値 特に規定しない

解説 $*self*->{before}および$*self*->{before_in_job}の各関数をジョブスクリプトから起動した Perl プロセスで実行するための Perl スクリプトの内容を生成し，その結果を改行文字で区切った文字列配列への参照として$*self*->{before_in_job_script}に格納する．

生成される Perl スクリプトについて，以下の要件をみたさなければならない．

- この Perl スクリプトは，$*self*に対する builtin::submit 関数の適用時の関数$*self*->{before}および$*self*->{before_in_job}を呼び出す．ただし，$*self*->{before}の呼び出しは，$*self*->{before_to_job}の値が真のときのみ行う．
- 上記の関数の本体においては，以下のものがグローバル変数，トップレベル関数として参照できる．
  - $*self*が指すジョブオブジェクトから，$*self*->{not_transfer_info}の参照先の配列に含まれる文字列に対応するメンバを取り除いたものへの参照が上記の関数本体においてグローバル変数$user::self を用いて参照可能．
  - $user::TEMPLATE{transfer_variable}の参照先の配列 が文字列'$*var*' を含む場合，Xcrypt で定義されたグローバル変数$user::*var* が，上記の関数本体においてグローバル変数$user::*var* として参照可能．
  - $user::TEMPLATE{transfer_variable}の参照先の配列 が文字列'@*var*' を含む場合，Xcrypt で定義されたグローバル変数@user::*var* が，上記の関数本体においてグローバル変数@user::*var* として参照可能．
  - $user::TEMPLATE{transfer_variable}の参照先の配列 が文字列'%*var*' を含む場合，Xcrypt で定義されたグローバル変数%user::*var* が，上記の関数本体においてグローバル変数%user::*var* として参照可能．
  - $user::TEMPLATE{transfer_variable}の参照先の配列 が文字列'&*func*' を含む場合，Xcrypt で定義されたトップレベル関数 user::*func* が，上記の関数本体においてトップレベル関数 user::*func* として参照可能．

  ただし，これらの参照可能なデータに「参照値」が含まれる場合（ジョブオブジェクトへの参照を含む），参照の深さ d を上限として Xcrypt 上のデータがコピーされたものが参照先の値となる．ここで，d は以下で定まる値である．
  - $*self*に対する builtin::submit の適用時，$*self*->{transfer_reference_level}の値が設定されていた場合，その値．

– それ以外の場合，5.

また，これらの変数に対する値の更新は Xcrypt 自体には反映されない．

- 上記の関数の本体において利用可能なライブラリは実装依存である．
- $*self* に対応するジョブの状態が done に遷移した後，$*self*->{before}の返り値を$*self*->before_return()のメソッド呼び出し，$*self*->{before_in_job}の返り値を$*self*->before_in_job_return()のメソッド呼び出しの返り値としてそれぞれ獲得できる（上記の関数の返り値がスカラコンテキストかリストコンテキストかで異なる場合，before_return や before_in_job_return のメソッドをスカラコンテキスト・リストコンテキストにおいて呼び出すことで，対応した返り値が得られる）．ただし，これらの返り値に「参照値」が含まれる場合は，上の説明と同様にコピーされたデータが当該の参照先のデータとなる．

8.2.8 make_exe_in_job_script メソッド

引数

- $*self*: ジョブオブジェクトへの参照

返り値 特に規定しない

解説 $*self*->{exe_in_job}関数をジョブスクリプトから起動した Perl プロセスで実行するための Perl スクリプトの内容を生成し，その結果を改行文字で区切った文字列配列への参照として$*self*->{exe_in_job_script}に格納する．

生成される Perl スクリプトについて，以下の要件をみたさなければならない．

- この Perl スクリプトは，$*self* に対する builtin::submit 関数の適用時の関数$*self*->{exe}を呼び出す．
- 上記の関数の本体において参照可能な変数および利用可能なライブラリに関する仕様は make_before_in_job が生成する Perl スクリプトと同様である．
- $*self* に対応するジョブの状態が done に遷移した後，$*self*->{exe}の返り値を$*self*->exe_return()のメソッド呼び出しの返り値として獲得できる．スカラ，リストコンテキストの扱いや，返り値に「参照値」が含まれる場合の扱いは make_before_in_job と同様である．

8.2.9 make_after_in_job_script メソッド

引数

- $*self*: ジョブオブジェクトへの参照

返り値 特に規定しない

解説　$*self*->{after}および$*self*->{after_in_job}の各関数をジョブスクリプトから起動したPerlプロセスで実行するためのPerlスクリプトの内容を生成し，その結果を改行文字で区切った文字列配列への参照として$*self*->{after_in_job_script}に格納する．

生成されるPerlスクリプトについて，以下の要件をみたさなければならない．

- このPerlスクリプトは，$*self*に対するbuiltin::submit関数の適用時の関数$*self*->{after}および$*self*->{after_in_job}を呼び出す．ただし，$*self*->{after}の呼び出しは，$*self*->{after_to_job}の値が真のときのみ行う．
- 上記の関数の本体において参照可能な変数および利用可能なライブラリに関する仕様はmake_before_in_jobが生成するPerlスクリプトと同様である．
- $*self*に対応するジョブの状態がdoneに遷移した後，$*self*->{after}の返り値を$*self*->after_return()のメソッド呼び出し，$*self*->{after_in_job}の返り値を$*self*->after_in_job_return()のメソッド呼び出しの返り値としてそれぞれ獲得できる．スカラ，リストコンテキストの扱いや，返り値に「参照値」が含まれる場合の扱いはmake_before_in_jobと同様である．

9　コマンドラインインターフェース