Open in Colab

RedisでmmapしたファイルからDoubleの値を読み込むモジュールをGoogle Colabで作ってみた

Redisでは既存のファイルを読み込む機能が見当たりません。（寡聞にして知らないだけかもしれないので、知っていたら教えてください。）そこで、doubleの値を書き込んだファイルをmmapでマップして、そこから指定した位置の値を読み書きするモジュールを作ってみました。セルを順に実行するだけで試せるようコードの全量を記載しています。Redisのモジュールの作り方としても参考にしてください。

このモジュールを導入すると以下のコマンドが使えるようになります。

```
// file_pathにあるファイルをkeyに結びつける
MMAP key file_path

// keyからindex位置にある値を取得する
VGET key index

// keyから複数のindex位置にある値を取得する
VMGET key index [index ...]

// keyのindex位置に値を書き込む
VSET key index value [index value ...]

// keyにvalueを追加する
VADD key value [value ...]

// keyの最後の値を取得して削除する
VPOP key

// keyの値の数を取得する
VCOUNT key

// keyの内容を消去する
VCLEAR key
```

まずはRedisのソースを取得して、モジュールのディレクトリに移動します。

```
%cd /content
!wget https://download.redis.io/redis-stable.tar.gz
!tar -xzf redis-stable.tar.gz
%cd /content/redis-stable/src/modules
```

試しにサンプルのコードをmakeしてみましょう。

```
!make
```

モジュールの.soファイルができたか確認します。

```
!ls *.so
```

```
helloacl.so    hellocluster.so  hellohook.so   hellotype.so
helloblock.so  hellodict.so hellotimer.so  helloworld.so
```

それではfmmap.cにコードを書いていきましょう。読み書きする値はdoubleです。なおコードを見やすくするためにエラー処理は省いています。お試しになる場合は意地悪なコマンドを投入しないようお願いします😏

また、こちらで公開しているコードでは読み書きする値の型を選べるようになっていて、エラー処理も完備しております。

最初に必要なヘッダーのインクルードです。便利マクロも定義します。

```
%%writefile fmmap.c

#pragma GCC diagnostic ignored "-Wunused-parameter"

#include <sys/mman.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdbool.h>
#include <stdint.h>
#include <strings.h>
#include <string.h>

#include "../redismodule.h"
#include "../sds.h"
#include "../zmalloc.h"

// (RedisModuleString *)と(char *)を比較するマクロ
static inline int mstringcmp(const RedisModuleString *rs1, const char *s2)
{
  return strcasecmp(RedisModule_StringPtrLen(rs1, NULL), s2);
}

int ftruncate(int fildes, off_t length); // unistd.hにあるはずだがwarningが出るので
```

MMapObjectを定義します。mmapに必要な情報を詰め込みました。sdsはRedis内で使われる文字列型です。

```
%%writefile -a fmmap.c

typedef struct _MMapObject
{
  sds file_path;
  int fd;
  void *mmap;
  size_t file_size;
} MMapObject;
```

MMap型を保持する変数と、MMapObjectを生成する関数です。

```
%%writefile -a fmmap.c

RedisModuleType *MMapType = NULL; // MMap型を保持する変数

// MMapObjectの生成
MMapObject *MCreateObject(void)
{
  return (MMapObject *)zcalloc(sizeof(MMapObject));
}
```

MMapObjectを解放する関数です。

```
%%writefile -a fmmap.c

// MMapObjectの解放
void MFree(void *value)
{
  if (value == NULL) return;
  const MMapObject *obj_ptr = value;
  if (obj_ptr->mmap != NULL) munmap(obj_ptr->mmap, obj_ptr->file_size);
  if (obj_ptr->fd != -1) close(obj_ptr->fd);
  sdsfree(obj_ptr->file_path);
  zfree(value);
}
```

file_pathで指定したファイルをkeyにマッピングする関数です。

```
%%writefile -a fmmap.c

// MMAP key file_path
int MMap_RedisCommand(RedisModuleCtx *ctx, RedisModuleString **argv, int argc)
```

```
{
  RedisModule_AutoMemory(ctx); /* Use automatic memory management. */

  if (argc != 3) return RedisModule_WrongArity(ctx);

  RedisModuleKey *key = RedisModule_OpenKey(ctx, argv[1], REDISMODULE_READ | REDISMO

  // keyの型を確認する
  int type = RedisModule_KeyType(key);
  if (type != REDISMODULE_KEYTYPE_EMPTY &&
      RedisModule_ModuleTypeGetType(key) != MMapType) {
    return RedisModule_ReplyWithError(ctx, REDISMODULE_ERRORMSG_WRONGTYPE);
  }

  // keyが空なら新たに作成してmmapする
  MMapObject *obj_ptr;
  if (type == REDISMODULE_KEYTYPE_EMPTY) {
    obj_ptr = MCreateObject();
    obj_ptr->file_path = sdsnew(RedisModule_StringPtrLen(argv[2], NULL));
    obj_ptr->fd = open(obj_ptr->file_path, O_RDWR | O_CREAT, 0666);
    if (obj_ptr->fd == -1) {
        MFree(obj_ptr);
        return RedisModule_ReplyWithError(ctx, sdsnew(RedisModule_StringPtrLen(argv[
    }

    struct stat sb;
    fstat(obj_ptr->fd, &sb);
    obj_ptr->file_size = sb.st_size;
    obj_ptr->mmap = mmap(NULL, sb.st_size, PROT_READ | PROT_WRITE, MAP_SHARED, obj_p
    RedisModule_ModuleTypeSetValue(key, MMapType, obj_ptr);
  }
  // 既存のkeyがある場合は同一性を確認する
  else {
    obj_ptr = RedisModule_ModuleTypeGetValue(key);
    if (obj_ptr == NULL) {
      RedisModule_ReplyWithNull(ctx);
      return REDISMODULE_ERR;
    }
    // 既存のファイルと異なる場合はエラー
    if (strcmp(obj_ptr->file_path, RedisModule_StringPtrLen(argv[2], NULL)) != 0) {
      return RedisModule_ReplyWithError(ctx, "It is already mapped on another file."
    }
  }

  return RedisModule_ReplyWithLongLong(ctx, obj_ptr->file_size / sizeof(double));
}
```

indexで指定する位置の値を取得する関数です。

```
%%writefile -a fmmap.c

// VGET key index
int VGet_RedisCommand(RedisModuleCtx *ctx, RedisModuleString **argv, int argc)
{
  RedisModule_AutoMemory(ctx); /* Use automatic memory management. */

  RedisModuleKey *key =
    RedisModule_OpenKey(ctx, argv[1], REDISMODULE_READ | REDISMODULE_WRITE);

  long long index;
  RedisModule_StringToLongLong(argv[2], &index);

  MMapObject *obj_ptr = RedisModule_ModuleTypeGetValue(key);

  // indexが範囲外ならNullを返す
  if (obj_ptr->file_size < (size_t)index * sizeof(double) || index < 0) {
    RedisModule_ReplyWithNull(ctx);
  }
  // mmap[index]を返す
  else {
    RedisModule_ReplyWithDouble(ctx, ((double*)obj_ptr->mmap)[index]);
  }
  return REDISMODULE_OK;
}
```

複数のindexで示した位置の値を取得する関数です。

```
%%writefile -a fmmap.c

// VMGET key index [index ...]
int VMGet_RedisCommand(RedisModuleCtx *ctx, RedisModuleString **argv, int argc)
{
  RedisModule_AutoMemory(ctx); /* Use automatic memory management. */

  RedisModuleKey *key =
      RedisModule_OpenKey(ctx, argv[1], REDISMODULE_READ | REDISMODULE_WRITE);

  MMapObject *obj_ptr = RedisModule_ModuleTypeGetValue(key);

  RedisModule_ReplyWithArray(ctx, argc - 2);
  for (int i = 2; i < argc; i++) {
    long long index;
    RedisModule_StringToLongLong(argv[i], &index);
    // indexが範囲外ならNullを返す
    if (obj_ptr->file_size < (size_t)index * sizeof(double) || index < 0) {
      RedisModule_ReplyWithNull(ctx);
    }
    // mmap[index]を返す
    else {
      RedisModule_ReplyWithDouble(ctx, ((double*)obj_ptr->mmap)[index]);
    }
  }
  return REDISMODULE_OK;
}
```

indexで示した位置に値を書き込む関数です。

```
%%writefile -a fmmap.c

// VSET key index value [index value ...]
int VSet_RedisCommand(RedisModuleCtx *ctx, RedisModuleString **argv, int argc)
{
  RedisModule_AutoMemory(ctx); /* Use automatic memory management. */

  RedisModuleKey *key =
      RedisModule_OpenKey(ctx, argv[1], REDISMODULE_READ | REDISMODULE_WRITE);

  MMapObject *obj_ptr = RedisModule_ModuleTypeGetValue(key);

  long long index;
  double value;
  // 書き込む値の型を確認する
  for (int i = 3; i < argc; i += 2) {
    if (RedisModule_StringToDouble(argv[i], &value) == REDISMODULE_ERR) {
      return RedisModule_ReplyWithError(ctx, "value must be double.");
    }
  }
  int n_factors = 0;  // 書き込めた要素の数を保持する変数
  // mmap[index]に値を書き込む
  for (int i = 2; i < argc; i += 2) {
    RedisModule_StringToLongLong(argv[i], &index);
    // indexが範囲内の時のみ書き込む
    if (0 <= index && (size_t)index * sizeof(double) < obj_ptr->file_size) {
      RedisModule_StringToDouble(argv[i + 1], &value);
      ((double*)obj_ptr->mmap)[index] = (double)value;
      ++n_factors;
    }
  }
  // 書き込んだ回数を返す
  return RedisModule_ReplyWithLongLong(ctx, n_factors);
}
```

ファイルの末尾に値を追加する関数です。

```
%%writefile -a fmmap.c

// VADD key value [value ...]
int VAdd_RedisCommand(RedisModuleCtx *ctx, RedisModuleString **argv, int argc)
{
  RedisModule_AutoMemory(ctx); /* Use automatic memory management. */

  RedisModuleKey *key =
      RedisModule_OpenKey(ctx, argv[1], REDISMODULE_READ | REDISMODULE_WRITE);

  MMapObject *obj_ptr = RedisModule_ModuleTypeGetValue(key);
  double value;
  // 追加する値の型を確認する
  for (int i = 2; i < argc; ++i) {
    if (RedisModule_StringToDouble(argv[i], &value) == REDISMODULE_ERR) {
      return RedisModule_ReplyWithError(ctx, "value must be double.");
    }
  }
  // mmapを拡張して書き込む
  size_t new_size = obj_ptr->file_size + sizeof(double) * (argc - 2);
  ftruncate(obj_ptr->fd, new_size);
  munmap(obj_ptr->mmap, obj_ptr->file_size);
  obj_ptr->mmap = mmap(NULL, new_size, PROT_READ | PROT_WRITE, MAP_SHARED, obj_ptr->
  for (int i = 2; i < argc; ++i) {
    size_t index = obj_ptr->file_size / sizeof(double);
    RedisModule_StringToDouble(argv[i], &value);
    obj_ptr->file_size += sizeof(double);
    ((double*)obj_ptr->mmap)[index] = (double)value;
  }

  // 書き込んだ要素の数を返す
  return RedisModule_ReplyWithLongLong(ctx, argc - 2);
}
```

ファイルに含まれる値の数を取得する関数です。

```
%%writefile -a fmmap.c

// VCOUNT key
int VCount_RedisCommand(RedisModuleCtx *ctx, RedisModuleString **argv, int argc)
{
  RedisModule_AutoMemory(ctx); /* Use automatic memory management. */

  RedisModuleKey *key =
      RedisModule_OpenKey(ctx, argv[1], REDISMODULE_READ | REDISMODULE_WRITE);

  MMapObject *obj_ptr = RedisModule_ModuleTypeGetValue(key);

  return RedisModule_ReplyWithLongLong(ctx, obj_ptr->file_size / sizeof(double));
}
```

ファイルの内容を消去する関数です。

```
%%writefile -a fmmap.c

// VCLEAR key
int VClear_RedisCommand(RedisModuleCtx *ctx, RedisModuleString **argv, int argc)
{
  RedisModule_AutoMemory(ctx); /* Use automatic memory management. */

  RedisModuleKey *key =
      RedisModule_OpenKey(ctx, argv[1], REDISMODULE_READ | REDISMODULE_WRITE);

  MMapObject *obj_ptr = RedisModule_ModuleTypeGetValue(key);

  munmap(obj_ptr->mmap, obj_ptr->file_size);
  ftruncate(obj_ptr->fd, 0);
  obj_ptr->mmap = mmap(NULL, 0, PROT_READ | PROT_WRITE, MAP_SHARED, obj_ptr->fd, 0);
  RedisModule_ReplyWithLongLong(ctx, obj_ptr->file_size / sizeof(double));
  obj_ptr->file_size = 0;

  return REDISMODULE_OK;
}
```

ファイルの末尾から数値を取得して、削除する関数です。

```
%%writefile -a fmmap.c

// VPOP key
int VPop_RedisCommand(RedisModuleCtx *ctx, RedisModuleString **argv, int argc)
{
  RedisModule_AutoMemory(ctx); /* Use automatic memory management. */

  RedisModuleKey *key =
      RedisModule_OpenKey(ctx, argv[1], REDISMODULE_READ | REDISMODULE_WRITE);

  MMapObject *obj_ptr = RedisModule_ModuleTypeGetValue(key);

  if (obj_ptr->file_size == 0) {
    RedisModule_ReplyWithNull(ctx);
  }
  else {
    size_t index = obj_ptr->file_size / sizeof(double) - 1;
    RedisModule_ReplyWithDouble(ctx, ((double*)obj_ptr->mmap)[index]);
    munmap(obj_ptr->mmap, obj_ptr->file_size);
    obj_ptr->file_size -= sizeof(double);
    ftruncate(obj_ptr->fd, obj_ptr->file_size);
    obj_ptr->mmap = mmap(NULL, obj_ptr->file_size, PROT_READ | PROT_WRITE, MAP_SHARE
  }
  return REDISMODULE_OK;
}
```

RedisのRDBファイルにmmapに関する情報を保存する関数です。

```
%%writefile -a fmmap.c

void MRdbSave(RedisModuleIO *rdb, void *value)
{
  MMapObject *obj_ptr = value;
  RedisModule_SaveStringBuffer(rdb, obj_ptr->file_path, sdslen(obj_ptr->file_path));
  msync(obj_ptr->mmap, obj_ptr->file_size, MS_ASYNC);
}
```

RedisのRDBファイルからmmapに関する情報を読み出す関数です。

```
%%writefile -a fmmap.c

void *MRdbLoad(RedisModuleIO *rdb, int encver)
{
  MMapObject *obj_ptr = MCreateObject();
  obj_ptr->file_path = sdsnew(RedisModule_StringPtrLen(RedisModule_LoadString(rdb),
  obj_ptr->fd = open(obj_ptr->file_path, O_RDWR);

  struct stat sb;
  fstat(obj_ptr->fd, &sb);
  obj_ptr->file_size = sb.st_size;
  obj_ptr->mmap = mmap(NULL, sb.st_size, PROT_READ | PROT_WRITE, MAP_SHARED, obj_ptr

  return obj_ptr;
}
```

RedisのAOFを利用するための関数です。

```
%%writefile -a fmmap.c

void MAofRewrite(RedisModuleIO *aof, RedisModuleString *key, void *value)
{
  char buffer[0x200];
  MMapObject *obj_ptr = (MMapObject*)value;
  RedisModule_EmitAOF(aof, "MMAP", "sc",
                      key,
                      obj_ptr->file_path);
  RedisModule_EmitAOF(aof, "MCLEAR", "ss", key, obj_ptr->file_path);
  for (size_t i = 0; i < obj_ptr->file_size; i += sizeof(double)) {
    double value = *(double *)((uint8_t*)obj_ptr->mmap + i);
    sprintf(buffer, "%.16f", value);
    RedisModule_EmitAOF(aof, "MADD", "sbc", key, buffer);
  }
}
```

その他、Redisのモジュールに必要な関数です。

```
%%writefile -a fmmap.c

size_t MMemUsage(const void *value)
{
  const MMapObject *obj_ptr = value;
  return obj_ptr->file_size;
}

void MDigest(RedisModuleDigest *md, void *value)
{
  REDISMODULE_NOT_USED(md);
  REDISMODULE_NOT_USED(value);
}
```

Redisのコマンドを作成するマクロ

```
%%writefile -a fmmap.c

#define CREATE_CMD(name, tgt, attr, key_pos, key_last)                    \
  do {                                                                    \
    if (RedisModule_CreateCommand(ctx, name, tgt, attr, key_pos, key_last, \
                                  1) != REDISMODULE_OK) {                 \
      return REDISMODULE_ERR;                                             \
    }                                                                     \
  } while (0);
```

モジュールのロード時に呼ばれて、モジュールを準備する関数です。ここで各コマンドとそれを実行する関数を結びつけます。

```
%%writefile -a fmmap.c

int RedisModule_OnLoad(RedisModuleCtx *ctx, RedisModuleString **argv, int argc)
{
  REDISMODULE_NOT_USED(argv);
  REDISMODULE_NOT_USED(argc);

  RedisModule_Init(ctx, "FuchiMMap", 1, REDISMODULE_APIVER_1);

  RedisModuleTypeMethods tm = {.version = REDISMODULE_TYPE_METHOD_VERSION,
                               .rdb_load = MRdbLoad,
                               .rdb_save = MRdbSave,
                               .aof_rewrite = MAofRewrite,
                               .mem_usage = MMemUsage,
                               .free = MFree,
                               .digest = MDigest};

  MMapType = RedisModule_CreateDataType(ctx, "FuchiMMap", 0, &tm);

  // MMAP key file_path
  CREATE_CMD("MMAP", MMap_RedisCommand, "write fast", 1, 1);

  // VCLEAR key
  CREATE_CMD("VCLEAR", VClear_RedisCommand, "write fast", 1, 1);

  // VADD key value [value ...]
  CREATE_CMD("VADD", VAdd_RedisCommand, "write fast", 1, 1);

  // VGET key index
  CREATE_CMD("VGET", VGet_RedisCommand, "readonly fast", 1, 1);

  // VMGET key index [index ...]
  CREATE_CMD("VMGET", VMGet_RedisCommand, "readonly fast", 1, 1);

  // VSET key index value [index value ...]
  CREATE_CMD("VSET", VSet_RedisCommand, "write fast", 1, 1);

  // VCOUNT key
  CREATE_CMD("VCOUNT", VCount_RedisCommand, "readonly fast", 1, 1);

  // VPOP key
  CREATE_CMD("VPOP", VPop_RedisCommand, "write fast", 1, 1);

  return REDISMODULE_OK;
}
```

ソースをビルドするためのMakefileです。

```
%%writefile Makefile.fmmap

SHOBJ_CFLAGS ?= -W -Wall -fno-common -g -ggdb -std=c99 -O2
SHOBJ_LDFLAGS ?= -shared

.SUFFIXES: .c .so .xo .o

all: fmmap.so

.c.xo:
    $(CC) -I. $(CFLAGS) $(SHOBJ_CFLAGS) -fPIC -c $< -o $@

fmmap.xo: ../redismodule.h

fmmap.so: fmmap.xo
    $(LD) -o $@ $^ $(SHOBJ_LDFLAGS) $(LIBS) -lc

clean:
    rm -rf *.xo *.so
```

モジュールをmakeします。

```
!make -f Makefile.fmmap
!ls fmmap.so
```

作成したモジュールを読み込めるように設定ファイルを準備します。

```
!cp ../../redis.conf .
```

```
%%writefile -a redis.conf
enable-module-command yes
loadmodule /content/redis-stable/src/modules/fmmap.so
```

Redisをインストールします。

```
!sudo yes | add-apt-repository ppa:redislabs/redis
!sudo apt-get update
!sudo apt-get install redis
```

設定ファイルに書き込みます。

```
%%writefile -a /etc/redis/redis.conf
enable-module-command yes
loadmodule /content/redis-stable/src/modules/fmmap.so
```

Redisを実行します。

```
!service redis-server start
```

Redisが実行されているか確認します。

```
!sleep 1
!ps aux | grep redis | grep -v grep
```

```
redis       3195  0.0  0.0  59132  6380 ?        Ssl  01:21   0:00 /usr/bin/redis-se
```

ファイルを書き込む場所を準備します。

```
!mkdir /content/db
!chmod 777 /content/db
```

dbにファイルをマップします。戻り値は数値の数で、新規なので0になります。

```
!echo "MMAP db /content/db/file.mmap" | redis-cli
```

```
(integer) 0
```

file.mmapファイルができていることを確認します。まだファイルサイズは0です。

```
!ls -l /content/db
```

```
total 0
-rw------- 1 redis redis 0 Jun 20 01:21 file.mmap
```

値を追加してみます。戻り値は追加した数値の数です。

```
!echo "VADD db 0.0" | redis-cli
```

```
(integer) 1
```

ファイルサイズが8バイトに増えているのが確認できます。

```
!ls -l /content/db
```

```
total 4
-rw------- 1 redis redis 8 Jun 20 01:21 file.mmap
```

コマンドをファイルに書き出して実行してみましょう。

```
%%writefile command
vadd db 0.1
vadd db 0.2
vadd db 0.3
vadd db 0.4
vadd db 0.5
vadd db 0.6
vadd db 0.7
vadd db 0.8
vadd db 0.9
vcount db
```

実行すると、登録数が10になっているのがわかります。

```
!redis-cli < command
```

```
(integer) 1
(integer) 1
(integer) 1
(integer) 1
(integer) 1
(integer) 1
(integer) 1
(integer) 1
(integer) 1
(integer) 10
```

ファイルも80バイトに増えています。

```
!ls -l /content/db
```

```
total 4
-rw------- 1 redis redis 80 Jun 20 01:21 file.mmap
```

ひとつのコマンドで複数追加もできます。追加した値の数が返ります。

```
!echo "VADD db 1.0 1.1 1.2 1.3 1.4 1.5" | redis-cli
```

```
(integer) 6
```

値を取り出してみます。

```
!echo "VGET db 5" | redis-cli
```

```
"0.5"
```

複数の場合はこうなります。浮動小数点なので誤差があります。

```
!echo "VMGET db 1 2 3 4 5" | redis-cli
```

```
1) "0.10000000000000001"
2) "0.20000000000000001"
3) "0.29999999999999999"
4) "0.40000000000000002"
5) "0.5"
```

値を変更することもできます。db[5]に50、db[10]に100を設定します。VSETの戻り値は変更した箇所の数です。

```
!echo "VSET db 5 50 10 100" | redis-cli
!echo "VMGET db 5 10" | redis-cli
```

```
(integer) 2
1) "50"
2) "100"
```

末尾の値を取り出して削除します。

```
!echo "VCOUNT db" | redis-cli
!echo "VGET db 15" | redis-cli
!echo "VPOP db" | redis-cli
!echo "VCOUNT db" | redis-cli
```

```
(integer) 16
"1.5"
"1.5"
(integer) 15
```

Redisを停止して再起動します。

```
!service redis-server stop
!service redis-server start
```

再起動しても値は残っています。

```
!echo "VCOUNT db" | redis-cli
```

```
(integer) 15
```

dbを削除しても再度マッピングすればファイルは元のままなので値を取得できます。

```
!echo "KEYS *" | redis-cli
!echo "DEL db" | redis-cli
!echo "KEYS *" | redis-cli
!ls -l /content/db/
!echo "MMAP dba /content/db/file.mmap" | redis-cli
!echo "VGET dba 5" | redis-cli
```

```
1) "db"
(integer) 1
(empty array)
total 4
-rw------- 1 redis redis 120 Jun 20 01:21 file.mmap
(integer) 15
"50"
```

内容をクリアするにはこうします。

```
!echo "VCLEAR dba" | redis-cli
!echo "VCOUNT dba" | redis-cli
```

```
(integer) 15
(integer) 0
```

ファイルのサイズが0になります。

```
!ls -l /content/db/
```

```
total 0
-rw------- 1 redis redis 0 Jun 20 01:21 file.mmap
```

dbaを削除します。

```
!echo "DEL dba" | redis-cli
```

```
(integer) 1
```

Pythonで100個のdoubleを書き込んだファイルを作成します。

```
import struct
with open('/content/db/file.mmap', 'wb') as fout:
    for i in range(100):
        fout.write(struct.pack('d', i))
```

ファイルをマップして中身を確認します。

```
!echo "mmap db /content/db/file.mmap" | redis-cli
!echo "vcount db" | redis-cli
!echo "vmget db 1 3 5 7 9" | redis-cli
```

```
(integer) 100
(integer) 100
1) "1"
2) "3"
3) "5"
4) "7"
5) "9"
```

Redisを停止します。redis-serverは残っていません。

```
!echo "shutdown" | redis-cli
!sleep 1
!ps aux | grep redis
```

```
root        3338  0.0  0.0   6904  3172 ?        S    01:21   0:00 /bin/bash -c ps a
root        3340  0.0  0.0   6444   724 ?        S    01:21   0:00 grep redis
```

ファイルは残っています。

```
!ls -l /content/db
```

```
total 4
-rw------- 1 redis redis 800 Jun 20 01:21 file.mmap
```

以上です。