Sphere

Skeleton for PHysical and Engineering REsearch

Ver 1.0.0

spl_fio による SBX/SPX ファイル　ユーザーマニュアル

2010 年 8 月 01 日

目次

１．SBX/SPX ファイル読込・書込処理の概要

Sphere にてサポートされている SBX/SPX ファイルの読み込み、書き込みを行う低レベル関数の使用方法について、記述します。

必要ライブラリは以下です。

libsp.a　　低レベルライブラリ

以下の低レベル関数について使用方法を記述します。

No	SBX ファイル	SPX ファイル	説明
1	splReadSbx	splReadSpx	SBX/SPX ファイルからヘッダー情報、データを読み込みます。
2	splWriteSbx	splWriteSpx	SBX/SPX ファイルにデータを書き込みます。
3	splGetSbxInfo	splGetSpxInfo	SBX/SPX ファイルからヘッダー情報のみを取得します。

低レベル関数、構造体の詳細については、”SPL ライブラリ（低レベルライブラリ）　マニュアル”を参照してください。

２．SBX/SPX ファイルからの読込手順

SBX/SPX ファイルからの読み込みは、splReadSbx/splReadSpx 関数を使用します。

（SBX ファイル）

関数名	splReadSbx	SBX ファイルからの読み込みを行う。
引数	splSvFileList*file_list　[in]	読み込むファイルのリスト
	int mode　[in]	読込モード
	splSbx* obj　[in,out]	読込情報の splSbx 構造体
	int readRank　[in]	ノード識別子
	SPL_Comm grp　[in]	コミュニケーショングループ
戻り値	SplBool	成功：SplTrue、失敗：SplFalse

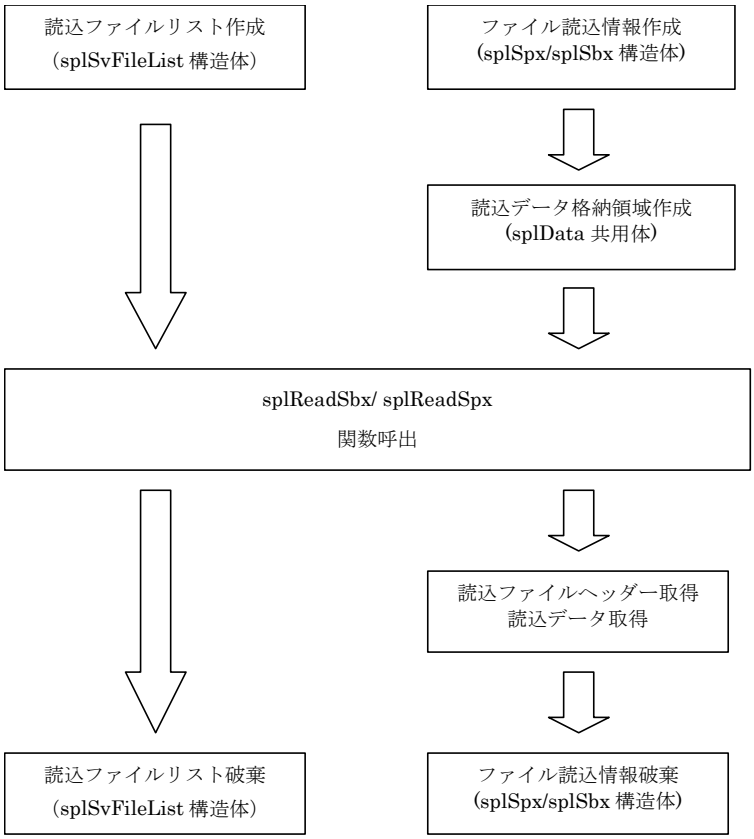
（SPX ファイル）

関数名	splReadSpx	SPX ファイルからの読み込みを行う
引数	splSvFileList*file_list　[in]	読み込むファイルのリスト
	int mode　[in]	読込モード
	splSpx* obj　[in,out]	読込情報の splSpx 構造体

	int readRank　[in]	ノード識別子
	SPL_Comm grp　[in]	コミュニケーショングループ
戻り値	SplBool	成功：SplTrue、失敗：SplFalse

しかし、関数の使用前に読込ファイルリスト（splSvFileList 構造体)、ファイル読込情報(splSpx/splSbx 構造体)を作成しなければなりません。

以下に、読込手順について記述します。



（１）読込ファイルリスト作成（splSvFileList 構造体）

（２）ファイル読込情報作成（splSbx/splSpx 構造体）

（３）読込データ格納領域作成（splData 共用体）

（４）splReadSbx/splReadSpx 関数呼出

（５）読込ファイルヘッダ、読込データ取得

（６）読込ファイルリスト破棄（splSvFileList 構造体）

（７）ファイル読込情報破棄（splSbx/splSpx 構造体）

以下に上記手順内容について説明します。

２．１　読込ファイルリスト作成（splSvFileList 構造体）

読込ファイルは、複数ファイルの追加が可能です。

この読込ファイルリストを splReadSbx/splReadSpx 関数に渡す為に splSvFileList 構造体

が用意されています。

splSvFileList 構造体へのファイル名の追加は以下の関数にて行います。

No	関数名	引数	説明
1	splAddSvFile		splSvFileList 構造体に読込 SBX/SPX ファイル名と始点グローバルインデックスを対で追加します。
		splSvFileList* obj	情報を追加する splSvFileList 構造体
		const char* fn	追加ファイル名
		const size_t sidx[3]	始点グローバルインデックス
2	splAddSvFileName		splSvFileList 構造体に読込 SBX/SPX ファイル名のみを追加します。 読込 SBX/SPX ファイルが拡張レコード：始点インデックスを持っている場合に使用します。
		splSvFileList* obj	情報を追加する splSvFileList 構造体
		const char* fn	追加ファイル名

（注意点）

読込 SBX/SPX ファイルが拡張レコード：始点インデックスを持っている場合、ファイルリストに追加した始点グローバルインデックスと異なる場合は、エラーとなります。

これは、読込対象の SBX/SPX ファイルがであるかの検証、整合性のチェックの為に始点インデックスが異なる場合はエラーとしています。

（使用例）

```
splSvFileList   fileList;                                                    (a)
// 初期化
if (splInitSvFileList(&fileList) != SplTrue) return SplFalse;               (b)
// ファイル名、始点インデックスの追加
const size_t sidx00 = {10, 20, 30};
if (splAddSvFile(&fileList, "sample00.sbx", sidx00)) != SplTrue) return SplFalse;   (c)
const size_t sidx01 = {100, 200, 300};
if (splAddSvFile(&fileList, "sample01.sbx", sidx01)) != SplTrue) return SplFalse;   (c)
```

(a) splSvFileList 構造体(読込ファイルリスト)を定義します。

(b) splInitSvFileList(...)関数により splSvFileList 構造体(読込ファイルリスト)を初期化します。

(c) splAddSvFile(...)関数によりファイル名、始点インデックスを追加します。
複数ファイルから読み込みを行う場合は、splAddSvFile を繰り返します

２．２　ファイル読込情報作成（splSbx/splSpx 構造体）

読み込みを行うガイドセル数、サイズ、始点インデックス、データ格納ポインタを設定する為に splSbx/splSpx 構造体（ファイル読込情報）を作成します。

（splSbx 構造体）

No	メンバー	説明	必須/不要	読込データへの反映
1	dims	次元数	必須	なし
2	vlen	ベクトル長(=1 固定)	必須(=1)	なし
3	dtype	**データ種別** １：char 型 ２：short 型 ４：int 型 実データのサイズ	必須	なし
4	gc	ガイドセル	必須	ガイドセルの設定
5	rlen	**実数データのサイズ** ４：float 型 ８：double 型 原点座標、ピッチのデータサイズ	不要	なし
6	crddef	定義位置	不要	なし
7	aux	AUX データ	不要	なし
8	size	サイズ	必須	サイズの設定
9	origin	原点座標	不要	なし
10	pitch	格子ピッチ	不要	なし
11	block_size	圧縮ブロックサイズ	不要	なし
12	wsize	拡張レコード： グローバルサイズ	不要	なし
13	start_idx	拡張レコード： 始点グローバルインデックス	必須	読込始点の設定
14	data	splData 共用体	必須	読込データ領域を確保しておく必要がある。

（注意）

"必須"項目は、読込ファイルと整合性が取れない場合はエラーとなります。

"不要"項目は、設定されていても、SBX ファイルのヘッダー情報と置き換わります。
"gc", "size", "start_idx"については、設定された値に従ってデータを読み込みます。

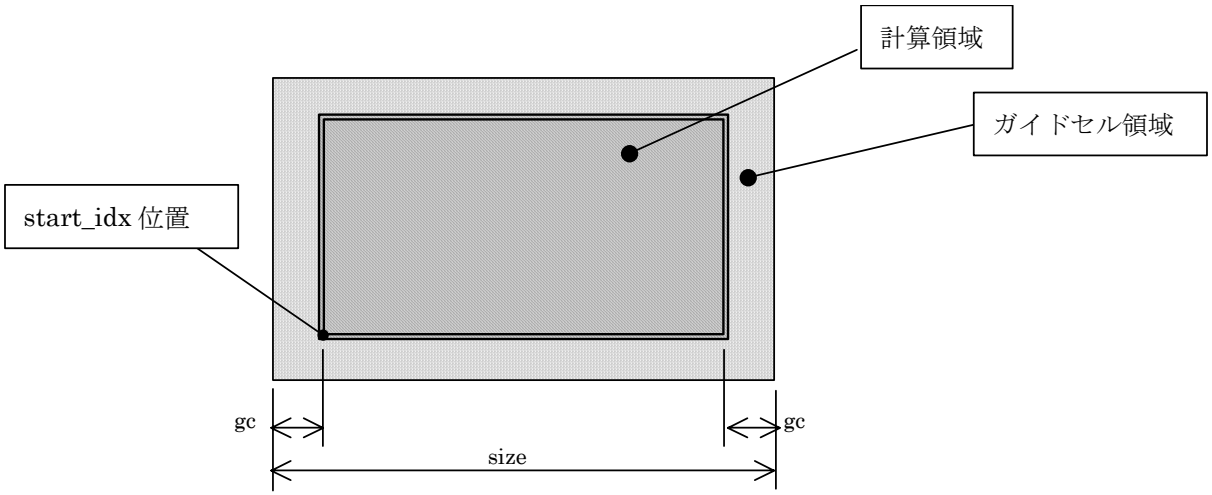
（splSpx 構造体）

No	メンバー	説明	必須/不要	読込データへの反映
1	dims	次元数	必須	なし
2	vlen	ベクトル長	必須	ベクトル長の設定
3	dtype	**データ種別** １：スカラーデータ ３：ベクトルデータ	必須	ベクトル長変更に伴うデータ種別の設定
4	gc	ガイドセル	必須	ガイドセルの設定
5	rlen	**実数データのサイズ** ４：float 型 ８：double 型 原点座標、ピッチ、実データのデータサイズ	不要	なし
6	crddef	定義位置	不要	なし
7	aux	AUX データ	不要	なし
8	size	サイズ	必須	サイズの設定
9	origin	原点座標	不要	なし
10	pitch	格子ピッチ	不要	なし
11	block_size	圧縮ブロックサイズ	不要	なし
12	wsize	拡張レコード： グローバルサイズ	不要	なし
13	start_idx	拡張レコード： 始点グローバルインデックス	必須	読込始点の設定
14	step	タイムステップ数	不要	なし
15	time	時刻	不要	なし
16	data	splData 共用体	必須	読込データ領域を確保しておく必要がある。

（注意）
"必須"項目は、読込ファイルと整合性が取れない場合はエラーとなります。
"不要"項目は、設定されていても、SBX ファイルのヘッダー情報と置き換わります。
"gc", "size", "start_idx", "vlen","dtype"については、設定された値に従ってデータを読み込みます。

dtype は vlen の設定値に対応した”１：スカラーデータ”又は”３：ベクトルデータ”を設定しなければなりません。

（SBX/SPX ファイル、splSbx/splSpx 構造体における注意点）



（１）サイズは、GC 込みである。

SBX/SPX ファイルヘッダ、及び splSbx/splSpx 構造体のサイズは、ガイドセル込みの値です。

データクラス等では、サイズは GC を含まない。

（２）start_idx 位置は、データ配列の{0,0,0}位置ではなく、gc 分内側の位置である。

（３）SBX と SPX のファイルヘッダ、、及び splSbx と splSpx 構造体以下の項目について異なります。

No	項目	SBX	SPX
1	vlen	ベクトル長(=1 固定)	ベクトル長（任意）
2	dtype	**データ種別** １：char 型 ２：short 型 ４：int 型 実データのデータサイズ	**データ種別** １：スカラーデータ ３：ベクトルデータ
3	rlen	**実数データのサイズ** ４：float 型 ８：double 型 原点座標、ピッチのデータサイズ	**実数データのサイズ** ４：float 型 ８：double 型 原点座標、ピッチ、実データのデータサイズ

4	１格子当りのデータサイズ	= dtype	= vlen × rlen
5	step time	なし	タイムステップ数 時刻

（使用例）

```
splSbx  sbx;                                                    (a)
// 初期化
if (splInitSbx(&sbx) != SplTrue) return SplFalse;               (b)
// 読込情報の設定
sbx.dims = 3;
sbx.vlen = 1;
sbx.dtype = 1;
sbx.gc = 1;
sbx.size[0] = 100; sbx.size[1] = 200; sbx.size[2] = 300;
sbx.start_idx[0] = 1; sbx.start_idx[1] = 2; sbx.start_idx[2] = 3;
```

(a) splSbx 構造体(ファイル読込情報)を定義します。

(b) splInitSbx(...)関数により splSbx 構造体(ファイル読込情報)を初期化します。

(c) 読込情報を設定します。

２．３　読込データ格納領域作成（splData 共用体）

読み込んだデータを格納する領域を作成します。

作成した領域は、splSbx/splSpx 構造体の data メンバーに設定します。

data メンバーは、splData 共用体となっています。

確保する領域サイズはファイル読込情報（splSbx/splSpx 構造体）と一致していなければなりません。

（SBX ファイルの場合）

確保する領域 ＝splSbx.size　x　splSbx.dtype

（SPX ファイルの場合）

確保する領域 ＝splSpx.size　x　splSpx.vlen　x　splSpx.rlen

（使用例）

```
splSbx sbx;                                                     (a)
（省略：初期化、サイズ等の設定）
```

```
size_t buf_size = sbx.dtype;                                    (b)
for (int i=0; i<sbx.dim;i++) buf_size = sbx.size[i];
sbx.data.c = (unsigned char*)malloc(buf_size);                  (c)
memset(sbx.data.c, 0x00, buf_size);                             (d)
```

(a) splSbx 構造体(ファイル読込情報)を定義します。

(b) 確保する領域サイズを算出します。。

(c) malloc 関数によりメモリ領域を確保し、splSbx.data 共用体に設定します。

(d) 確保したメモリ領域を初期化します。

２．４　splReadSbx/splReadSpx 関数呼出

splReadSbx/splReadSpx 関数を呼び出して、SBX/SPX ファイルからの読み込みを実行します。

関数名	splReadSbx	SBX ファイルからの読み込みを行う。
	splReadSpx	SPX ファイルからの読み込みを行う
引数	splSvFileList*file_list　[in]	読み込むファイルのリスト
	int mode　[in]	読込モード
	splSbx* obj　[in,out]	読込情報の splSbx 構造体 (splReadSbx の場合)
	splSpx* obj　[in,out]	読込情報の splSpx 構造体 (splReadSpx の場合)
	int readRank　[in]	ノード識別子
	SPL_Comm grp　[in]	コミュニケーショングループ
戻り値	SplBool	成功：SplTrue、失敗：SplFalse

以下、引数について説明します。

（１）splSvFileList* file_list　（入力）

読み込むファイルリストである splSvFileList 構造体を設定します。

前項の”読込ファイルリスト作成”を参照してください。

（２）int mode　（入力）

ファイルの読込モードを設定します。

モードによりファイルをオープンするノード、動作が異なります。

モード番号	説明
0	全てのノードがファイルをオープンして、ファイルを読み込みます。
1	readRank のノードがファイルをオープンし、読み込んだ値を他のノードにブロードキャストします。

（３）splSbx/splSpx* obj　（入力／出力）

ファイルの読込情報、データ領域を設定します。読み込み後ファイルヘッダ情報、読込データが返されます。

前項の”ファイル読込情報作成”、”読込データ格納領域作成”を参照してください。

（４）int readRank　（入力）

読込モード＝０の場合は設定の必要はありませんが、読込モード＝１の場合は読み込みを行うノード識別子を設定しなければなりません。

ノード識別子は、読込グループ番号内のローカルノード番号です。

（５）SPL_Comm grp　（入力）

読込モード＝０の場合は設定の必要はありませんが、読込モード＝１の場合は読み込みを行うコミュニケーショングループ番号を設定する必要があります。

（使用例）

```
splSvFileList file_list;                                          (a)
（省略：初期化、ファイル名の追加）
splSbx sbx;                                                       (b)
（省略：初期化、サイズ等の設定）
（省略：読込データ格納領域作成）
int mode = 1;
int readRank = 5;
SPL_Comm grp = SPL_COMM_WORLD;
if (splReadSbx(&file_list, mode, &sbx, readRank, grp) != SplTrue) {   (c)
    return SplFalse;
}
```

(a) splSvFileList 構造体(ファイルリスト)を定義、設定します。

(b) splSbx 構造体(ファイル読込情報)を定義、設定します。

(c) splReadSbx/splReadSpx 関数を呼び出して読み込みを実行します

２．５　読込ファイルヘッダ、読込データ取得

読み込みが正常終了した場合、ファイルヘッダ情報、データは splSbx/splSpx 構造体から取得できます。

情報	取得先
ファイルヘッダ情報	splSbx/splSpx 構造体メンバー

読込データ	splSbx/splSpx 構造体 . data メンバー

２．６　読込ファイルリスト破棄（splSvFileList 構造体）

ファイルリスト（splSvFileList 構造体）使用後、破棄処理を行わなければなりません。破棄の為に" splFinalizeSvFileList"関数を呼び出してください。

（使用例）

```
splSvFileList file_list;                                          (a)
（省略：初期化、ファイル名の追加）
if (splReadSbx(&file_list, mode, &sbx, readRank, grp) != SplTrue) {
    return SplFalse;
}
（省略：ファイルヘッダ、データ取得）
splFinalizeSvFileList(&file_list);                                (b)
```

(a) splSvFileList 構造体(ファイルリスト)を定義、設定します。

(b) splSvFileList 構造体を破棄します。

２．７　ファイル読込情報破棄（splSbx/splSpx 構造体）

ファイル読込情報（splSbx/splSpx 構造体）使用後、破棄処理を行わなければなりません。破棄の為に" splFinalizeSbx/ splFinalizeSpx"関数を呼び出してください。

関数名	splFinalizeSbx	splSbx 構造体の破棄を行う。。
	splFinalizeSpx	splSpx 構造体の破棄を行う。
引数	splSbx* obj	破棄を行う splSbx 構造体 (splFinalizeSbx の場合)
	splSpx* obj	破棄を行う splSpx 構造体 (splFinalizeSpx の場合)
	SplBool mem_free_flag	メモリ開放フラグ SplTrue:メモリ開放を行う SplFalse:メモリ開放を行わない。
戻り値	SplBool	成功：SplTrue、失敗：SplFalse

メモリ開放フラグは、splSbx/splSpx 構造体の data メンバーに確保した読込データ領域を開放するか、しないかのフラグです。

読込データを他の領域にコピーして、元の data メンバーの領域が必要ないのであれば開

放を行わなければなりません。

data メンバーの領域のポインタを使いまわすのであれば開放せず、使用先で開放してください。

（使用例）

```
splSbx sbx;                                                    (a)
（省略：初期化、サイズ等の設定）
（省略：読込データ格納領域作成）
if (splReadSbx(&file_list, mode, &sbx, readRank, grp) != SplTrue) {
      return SplFalse;
}
（省略：ファイルヘッダ、データ取得）
char* sbx_data = sbx.data.c                          // データポインタの取得
splFinalizeSbx (&sbx, SplFalse);                                   (b)
```

(a) splSbx 構造体(ファイル読込情報)を定義、設定します。

(b) splSbx 構造体をデータ領域は開放せずに破棄します。

３．SBX/SPX ファイルからの読込パラメータ

SBX/SPX ファイルからの読み込みにおいて読み込み領域の位置、サイズを変更するパラメータとして以下があります。

No	構造体	メンバー	説明
1	splSvFileList （ファイルリスト）	start_idx	ファイル始点インデックス
2	splSbx/splSpx 構造体 （ファイル読込情報）	gc	ガイドセル数
3		size	読込サイズ
4		start_idx	読込始点インデックス
5		vlen	SPX ファイルのみ ベクトル長

（注意点）

SBX/SPX ファイルからの読込み時に以下の「始点インデックス」を考慮しなければいけません。

No	格納場所	メンバー	説明	
1	splSvFileList 構造体 （ファイルリスト）	start_idx	ファイル始点インデックス	読込ファイルの始点インデックスを設定す

No	格納場所	メンバー	説明	
				る。
2	splSbx/splSpx 構造体（ファイル読込情報）	start_idx	読込始点インデックス	読込みを行う始点インデックスを設定する。
3	SBX/SPX ファイル::拡張レコード（任意）	start_idx	ファイル拡張レコード::始点インデックス	ファイルに設定済みの始点インデックス

splSbx/splSpx 構造体の start_idx（読込始点インデックス）は読込結果であるデータの始点インデックスですので任意の値を設定可能です。

しかし、SBX/SPX ファイル::拡張レコードの start_idx（始点インデックス）は任意の設定レコードですので splSvFileList 構造体の start_idx は SBX/SPX ファイル::拡張レコードの start_idx を補完するものです。

よって、以下の制限があります。

No	splSvFileList 構造体 ::start_idx	SBX/SPX ファイル::拡張レコード ::start_idx	設定規則
1	あり (splAddSvFile 関数により start_idx を設定)	あり (拡張レコードあり)	splSvFileList 構造体の start_idx は、SBX/SPX ファイル::拡張レコードの start_idx と同じ値でなければならない。
2	なし (splAddSvFileName 関数によりファイル名のみ設定)	あり (拡張レコードあり)	SBX/SPX ファイル::拡張レコードの start_idx を有効な始点インデックスとして読込みを行う。
3	あり (splAddSvFile 関数により start_idx を設定)	なし (拡張レコードなし)	splSvFileList::start_idx を有効な始点インデックスとして読込みを行う。
4	なし (splAddSvFileName 関数によりファイル名のみ設定)	なし (拡張レコードなし)	splSvFileList に登録ファイル数 １つの場合 start_idx={0,0,0}とする。とする。 ２つ以上の場合 始点インデックスが設定されていないのでエラー

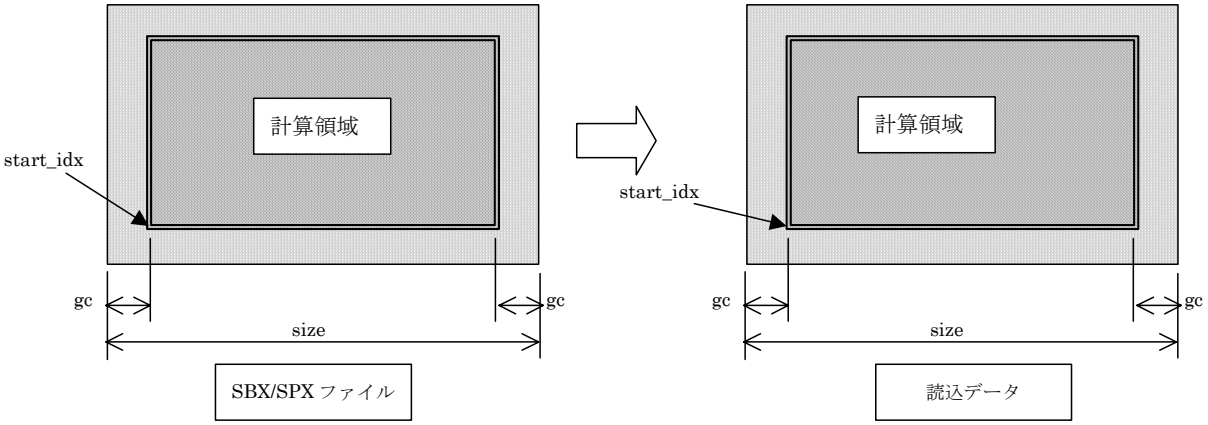
以下、上記パラメータの設定により読込データの位置、サイズがどのように変更されるか

記述します。

３．１　全領域読込

SBX/SPX ファイルのデータをすべて読み込む為には、サイズ、始点インデックスをファイルに記述されている通り設定しなければなりません。

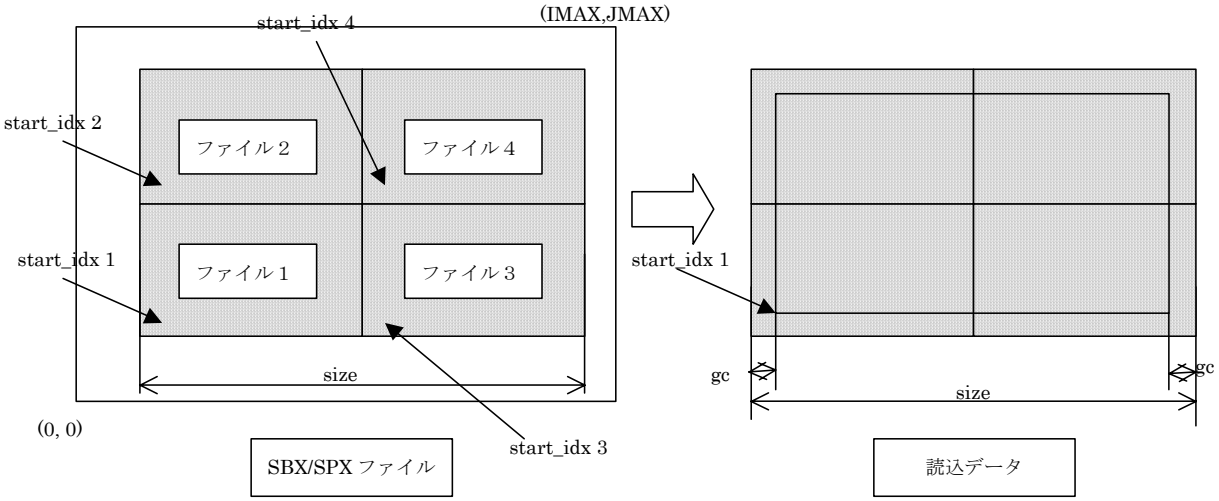
（単一ファイルからの読込）



No	構造体	メンバー	項目	変更内容
1	splSvFileList	start_idx	始点インデックス	ファイルの始点インデックスを設定する。 拡張レコードを持っている SBX/SPX ファイルでは拡張レコードの start_idx と同じ値を設定しなければならない。
2	splSbx/splSpx	gc	ガイドセル	読込ファイルヘッダのガイドセル数を設定する。
3		size	サイズ	読込ファイルヘッダのサイズを設定する。
4		start_idx	始点インデックス	読込ファイルリストに設定した start_idx と同じ値を設定しなければならない。 (=splSvFileList. start_idx)

（複数ファイルからの読込）

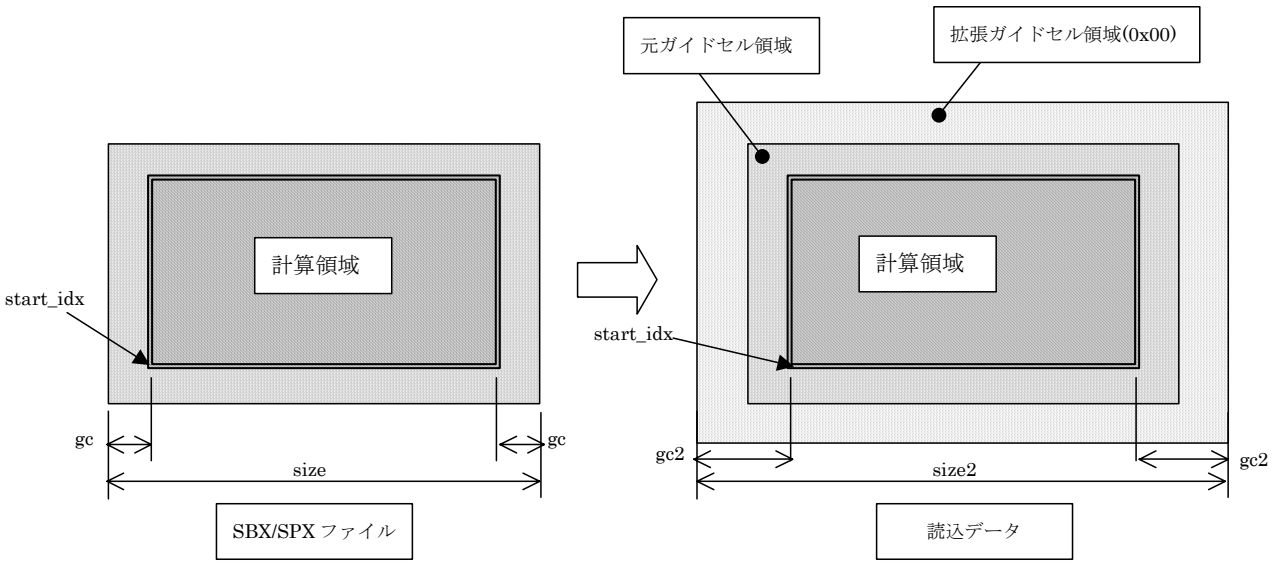
複数 SBX/SPX ファイルの読み込みを行う場合は、ファイルリストの最小の始点インデックスを設定し、最小始点インデックスから全領域を包括するサイズを設定しなければなりません。



No	構造体	メンバー	項目	変更内容
1	splSvFileList	start_idx	始点インデックス	ファイルの始点インデックスを設定する。 （例=start_idx1, start_idx2, start_idx3, start_idx4） 拡張レコードを持っている SBX/SPX ファイルでは拡張レコードの start_idx と同じ値を設定しなければならない。
2	splSbx/splSpx	gc	ガイドセル	読込ファイルヘッダのガイドセル数を設定する。
3		size	サイズ	複数ファイルのサイズを加算したサイズを設定する。
4		start_idx	始点インデックス	読込ファイルリストに設定した最小の start_idx を設定しなければならない。 (例=start_idx1)

３．２　ガイドセルの変更

ファイル読込情報のガイドセル数を設定することにより設定されたガイドセル数に変更して読込を行います。



計算領域を変更せずに、ガイドセル領域を拡張又は縮小する場合は、以下の値を変更しなければならない。

No	構造体	メンバー	項目	変更内容
1	splSvFileList	start_idx	始点インデックス	ファイルの始点インデックスを設定する。 拡張レコードを持っている SBX/SPX ファイルでは拡張レコードの start_idx と同じ値を設定しなければならない。
2	splSbx/splSpx	gc	ガイドセル	作成するデータのガイドセル数を設定する。 (=gc2)
3		size	サイズ	設定されたサイズの領域の読み込みを行う。 （単一ファイルですべてのデータを読み込む場合） SBX/SPX ファイルの size に変更分のガイドセル数を加算したサイズを設定しなければならない。 size2 = size + (gc2 – gc) * 2
4		start_idx	始点インデックス	読込ファイルリストに設定した start_idx と同じ値を設定しなければならない。 (=splSvFileList. start_idx)

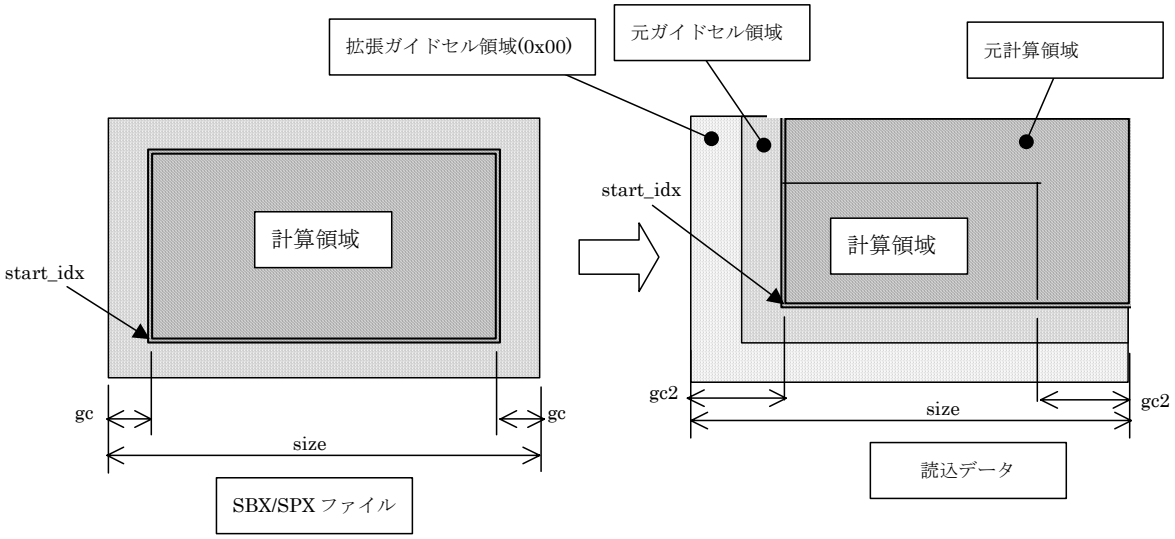
（例）

SBX/SPX ファイル　　: gc = 1, size = {6, 4, 7}

GC を 3 に変更　　: gc = 3, size = {10, 8, 11}

サイズを変更せず、ガイドセル値のみ変更した場合の読込データは以下のようになります。
（例）

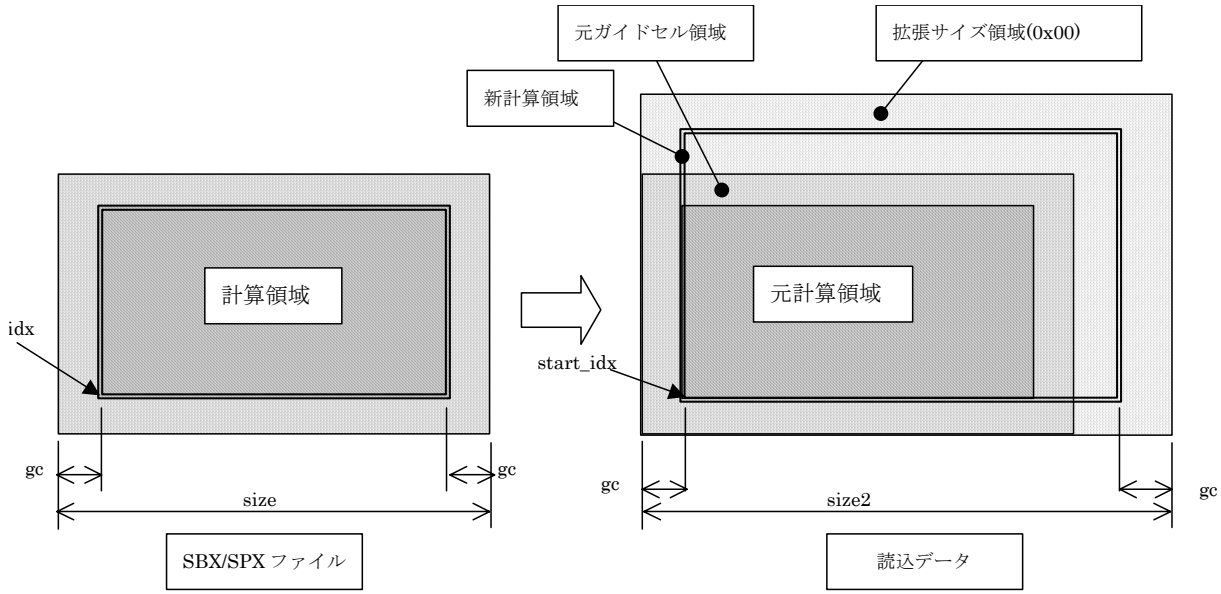
SBX/SPX ファイル　　：gc = 1, size = {6, 4, 7}
GC を 3 に変更　　：gc = 3, size = {6, 4, 7}



SBX/SPX ファイル及び splSbx/splSpx でのサイズはガイドセル込みの値であり、且つ始点インデックスの位置の関係から上記の様に計算領域が小さくなったデータとなる。

３．３　サイズの変更

ファイル読込情報のサイズを設定することにより設定されたサイズの領域の読込を行います。



SBX ファイルのサイズを拡張又は縮小する場合は、以下の値を変更しなければならない。

No	構造体	メンバー	項目	変更内容
1	splSvFileList	start_idx	始点インデックス	ファイルの始点インデックスを設定する。 拡張レコードを持っている SBX/SPX ファイルでは拡張レコードの start_idx と同じ値を設定しなければならない。
2	splSbx/splSpx	gc	ガイドセル	ガイドセルを変更しない場合は SBX/SPX ファイルの GC と同じ値を設定しなければならない。
3		size	サイズ	読込データのサイズを設定する。
4		start_idx	始点インデックス	読込ファイルリストに設定した start_idx と同じ値を設定しなければならない。 (=splSvFileList. start_idx)

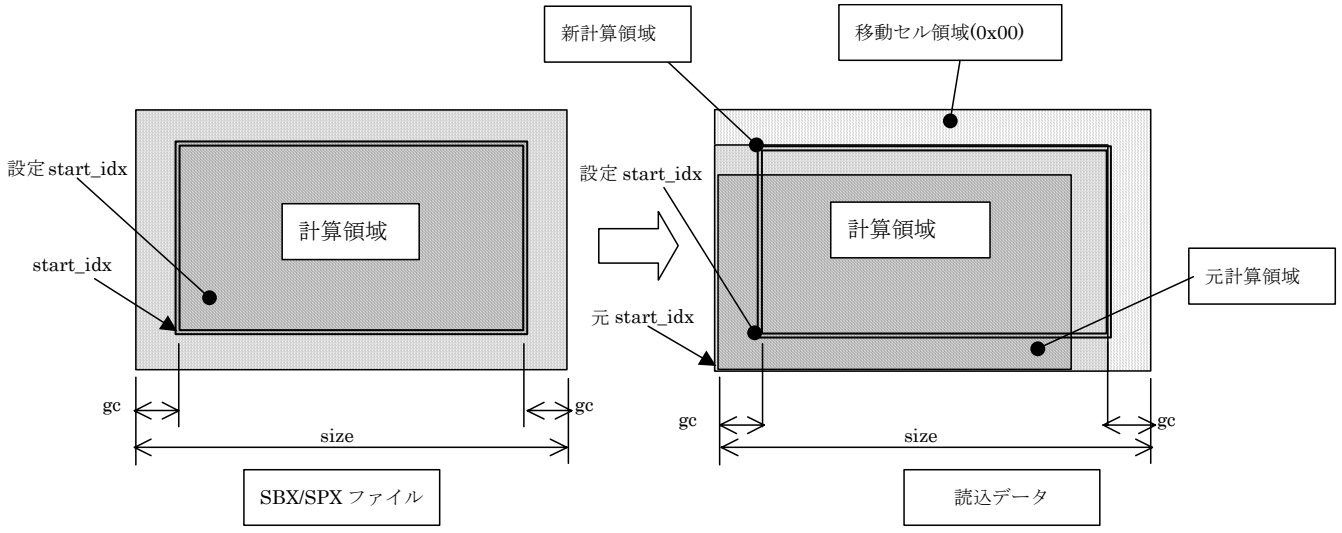
（例）

SBX/SPX ファイル　　：　size = {6, 4, 7}

i, k を 2 倍に変更　　：m_size = {12, 4, 14}

３．４　始点インデックスの変更

ファイル読込情報の始点インデックスを設定することにより設定された始点インデックスからの領域の読込を行います。



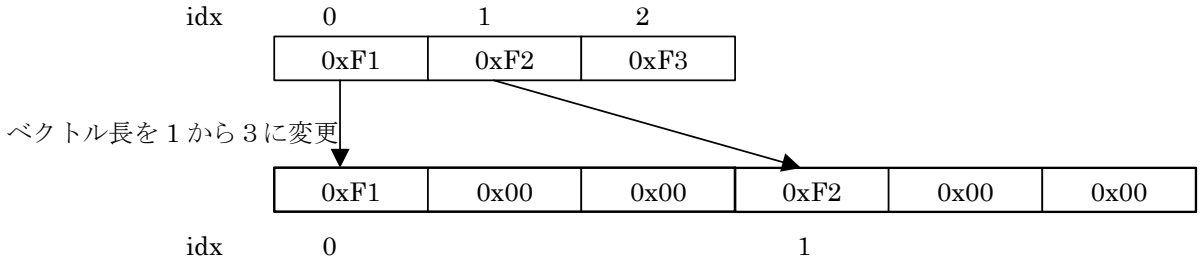
No	構造体	メンバー	項目	変更内容

1	splSvFileList	start_idx	始点インデックス	ファイルの始点インデックスを設定する。 拡張レコードを持っている SBX/SPX ファイルでは拡張レコードの start_idx と同じ値を設定しなければならない。
2	splSbx/splSpx	gc	ガイドセル	ガイドセルを変更しない場合は SBX/SPX ファイルの GC と同じ値を設定しなければならない。
3		size	サイズ	サイズを変更しない場合は SBX/SPX ファイルのサイズと同じ値を設定しなければならないが、始点インデックスの変更によりデータ領域を拡張、縮小する場合は、始点インデックスに合わせた変更が必要である。
4		start_idx	始点インデックス	読込ファイルリストに設定した start_idx を基準として読み込みを行う始点インデックスを設定する。

３．５　ベクトル長の変更（SPX ファイルのみ）

ファイル読込情報のベクトル長を設定することにより設定されたベクトル長にて読込データを作成します。

この機能は SPX ファイルのみです。（SBX ファイルのベクトル長は１に固定）



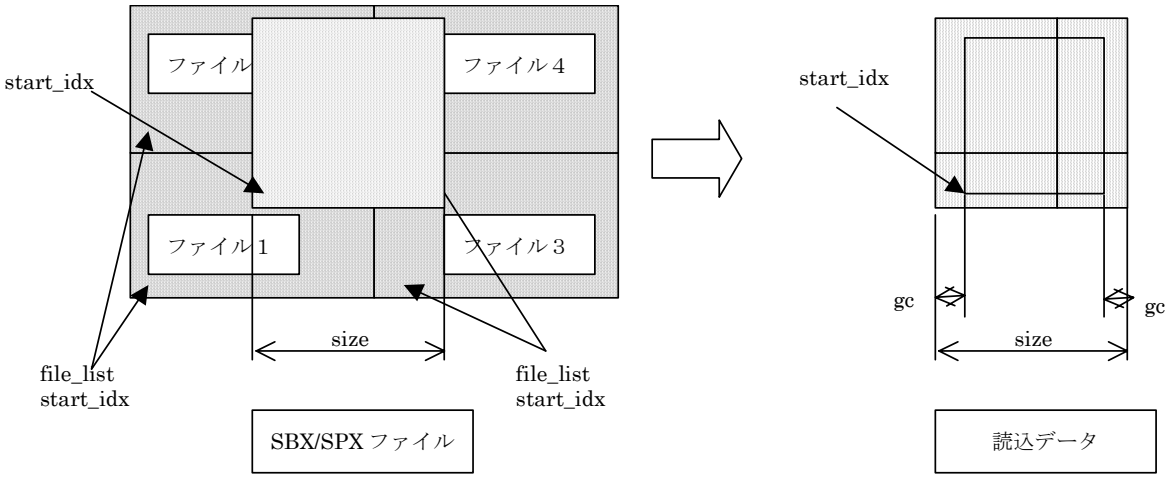
SPX ファイルのベクトル長を変更する場合は、以下の値を変更しなければならない。

No	構造体	メンバー	項目	変更内容
1	splSvFileList	start_idx	始点インデックス	ファイルの始点インデックスを設定する。 拡張レコードを持っている SBX/SPX ファイルでは拡張レコードの start_idx と同じ値を設定しなければならない。

2	splSbx/splSpx	gc	ガイドセル	読込ファイルヘッダのガイドセル数を設定する。
3		size	サイズ	読込ファイルヘッダのサイズを設定する。
4		start_idx	始点インデックス	読込ファイルリストに設定した start_idx と同じ値を設定しなければならない。 (=splSvFileList. start_idx)
5		vlen	ベクトル長	変更を行うベクトル長を設定する。
6		dtype	データ種別	ベクトル長の変更によりデータ種別が変わる場合は設定しなければならない。 １：スカラーデータ ３：ベクトルデータ

３．６　複数ファイルからの読み込み

ファイルリストと読込ヘッダーを設定することにより複数ファイルが構成する領域から任意の領域のデータを取得することができます。



複数 SBX/SPX ファイルの任意の位置からの領域を読み込む場合は、以下の値を変更しなければならない。

No	構造体	メンバー	項目	変更内容
1	splSvFileList	start_idx	始点インデックス	ファイルの始点インデックスを設定する。 拡張レコードを持っている SBX/SPX ファイルでは拡張レコードの start_idx と同じ値を設定しなければならない。

No	構造体	メンバー	項目	変更内容
2	splSbx/splSpx	gc	ガイドセル	ガイドセルを変更しない場合は SBX/SPX ファイルの GC と同じ値を設定しなければならない。
3		size	サイズ	読込データのサイズを設定する。
4		start_idx	始点インデックス	読込データの始点インデックスを設定する。

４．SBX/SPX ファイルへの書込手順

SBX/SPX ファイルへの書き込みは、splWriteSbx/splWriteSpx 関数を使用します。

（SBX ファイル）

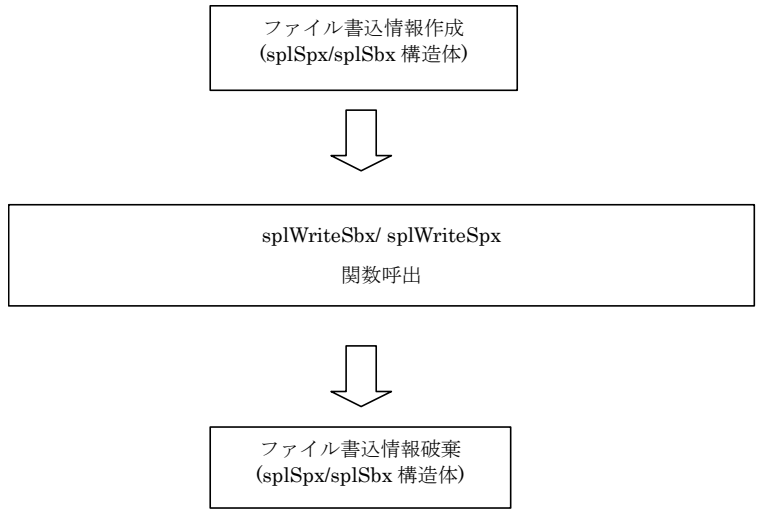
関数名	splWriteSbx	SBX ファイルへ書き込みを行う。
引数	const char* fname　[in]	書き込むファイル名
	int mode　[in]	書込モード
	SplBool cmp_flag	圧縮フラグ SplTrue:圧縮、SplFalse:非圧縮
	SplBool ext_rec_flag	拡張レコード出力フラグ SplTrue: 拡張レコードを出力する。 SplFalse: 拡張レコードを出力しない。
	size_t block_size	ブロックサイズ
	const splSbx* obj　[in]	書込情報の splSbx 構造体
	int writeRank　[in]	ノード識別子
	SPL_Comm grp　[in]	コミュニケーショングループ
戻り値	SplBool	成功：SplTrue、失敗：SplFalse

（SPX ファイル）

関数名	splWriteSpx	SPX ファイルへ書き込みを行う
引数	const char* fname　[in]	書き込むファイル名
	int mode　[in]	書込モード
	SplBool cmp_flag	圧縮フラグ SplTrue:圧縮、SplFalse:非圧縮
	SplBool ext_rec_flag	拡張レコード出力フラグ SplTrue: 拡張レコードを出力する。 SplFalse: 拡張レコードを出力しない。

	size_t block_size	ブロックサイズ
	splSpx* obj　[in,out]	書込情報の splSpx 構造体
	int writeRank　[in]	ノード識別子
	SPL_Comm grp　[in]	コミュニケーショングループ
戻り値	SplBool	成功：SplTrue、失敗：SplFalse

関数の使用前にファイル書込情報(splSpx/splSbx 構造体)を作成しなければなりません。
以下に、書込手順について記述します。



（1）ファイル書込情報作成（splSbx/splSpx 構造体）
（2）splWriteSbx/splWriteSpx 関数呼出
（3）ファイル書込情報破棄（splSbx/splSpx 構造体）
以下に上記手順内容について説明します。

４．１　ファイル書込情報作成（splSbx/splSpx 構造体）

書き込みを行う SBX/SPX ファイルのヘッダー情報を splSbx/splSpx 構造体（ファイル書込情報）に設定します。

（splSbx 構造体）

No	メンバー	説明	必須/不要	設定の注意点
1	dims	次元数	必須	なし
2	vlen	ベクトル長(=1 固定)	必須	１固定
3	dtype	**データ種別** １：char 型 ２：short 型 ４：int 型 実データのサイズ	必須	書込データ(data)のデータ型を設定

No	メンバー	説明	必須/不要	設定の注意点
4	gc	ガイドセル	必須	なし
5	rlen	**実数データのサイズ** ４：float 型 ８：double 型 原点座標、ピッチのデータサイズ	必須	なし
6	crddef	定義位置	必須	なし
7	aux	AUX データ	任意	なし
8	size	データサイズ	必須	書込データ(data)のサイズを設定
9	origin	原点座標	必須	なし
10	pitch	格子ピッチ	必須	なし
11	block_size	圧縮ブロックサイズ	任意	圧縮書込を行う場合は必須
12	wsize	拡張レコード： グローバルサイズ	任意	出力ファイルのサイズを設定します。
13	start_idx	拡張レコード： 始点グローバルインデックス	任意	書込データの始点インデックスを設定します。
14	data	splData 共用体	必須	書込データ

（splSpx 構造体）

No	メンバー	説明	必須/不要	設定の注意点
1	dims	次元数	必須	なし
2	vlen	ベクトル長	必須	書込データ(data)のベクトル長を設定
3	dtype	**データ種別** １：スカラーデータ ３：ベクトルデータ	必須	なし
4	gc	ガイドセル	必須	なし
5	rlen	**実数データのサイズ** ４：float 型 ８：double 型 原点座標、ピッチ、実データ	必須	書込データ(data)のデータ型を設定

No	メンバー	説明	必須/不要	設定の注意点
		のデータサイズ		
6	crddef	定義位置	必須	なし
7	aux	AUX データ	任意	なし
8	size	サイズ	必須	書込データ(data)のサイズを設定
9	origin	原点座標	必須	なし
10	pitch	格子ピッチ	必須	なし
11	block_size	圧縮ブロックサイズ	任意	圧縮書込を行う場合は必須
12	wsize	拡張レコード： グローバルサイズ	任意	出力ファイルのサイズを設定します。
13	start_idx	拡張レコード： 始点グローバルインデックス	任意	書込データの始点インデックスを設定します。
14	step	タイムステップ数	任意	なし
15	time	時刻	任意	なし
16	data	splData 共用体	必須	書込データ

書込データ(data：splData 共用体)には書き込むデータのポインタを、splData 共用体のメンバーに設定します。

（使用例）

```
splSbx   sbx;                                                    (a)
// 初期化
if (splInitSbx(&sbx) != SplTrue) return SplFalse;                (b)
// 書込情報の設定                                                 (c)
sbx.dims = 3;
sbx.vlen = 1;
sbx.dtype = 1;
sbx.gc = 1;
sbx.rlen = 4;
sbx.crddef = 2;
sbx.size[0] = 100; sbx.size[1] = 200; sbx.size[2] = 300;
sbx.origin [0] = 1.0; sbx.origin [1] = 2.0; sbx.origin [2] = 3.0;
sbx.pitch [0] = 1.0; sbx.pitch [1] = 1.0; sbx.pitch [2] = 1.0;
```

```
（一部省略）
sbx.data.c = buf;                                    (d)
```

(a) splSbx 構造体(ファイル書込情報)を定義します。

(b) splInitSbx(...)関数により splSbx 構造体(ファイル書込情報)を初期化します。

(c) 書込情報を設定します。

(d) 書込データのポインタを splData 共用体に設定します。

４．２　splWriteSbx/splWriteSpx 関数呼出

splWriteSbx/splWriteSpx 関数を呼び出して、SBX/SPX ファイルに書き込みを実行します。

関数名	splWriteSbx	SBX ファイルへ書き込みを行う。
	splWriteSpx	SPX ファイルへ書き込みを行う。
引数	const char* fname　[in]	書き込むファイル名
	int mode　[in]	書込モード
	SplBool cmp_flag	圧縮フラグ SplTrue:圧縮、SplFalse:非圧縮
	SplBool ext_rec_flag	拡張レコード出力フラグ SplTrue: 拡張レコードを出力する。 SplFalse: 拡張レコードを出力しない。
	size_t block_size	ブロックサイズ
	splSbx* obj　[in,out]	書込情報の splSbx 構造体 (splWriteSbx の場合)
	splSpx* obj　[in,out]	書込情報の splSpx 構造体 (splWriteSpx の場合)
	int readRank　[in]	ノード識別子
	SPL_Comm grp　[in]	コミュニケーショングループ
戻り値	SplBool	成功：SplTrue、失敗：SplFalse

以下、引数について説明します。

（１）const char* fname　（入力）

書き込むファイル名を指定します。

（２）int mode　（入力）

ファイルの書込モードを設定します。

モードによりファイルをオープンするノード、動作が異なります。

モード番号	説明
0	全てのノードがファイルをオープンして、ファイルを読み込みます。
1	writeRank のノードがファイルをオープンし、他のノードから情報を集めてデータを書き出します

（3）SplBool cmp_flag　（入力）

SBX/SPX ファイルのデータレコードを圧縮して書き込むか指定します。

SplTrue: データレコードをブロックサイズで圧縮して書き込む。

SplFalse:データレコードの圧縮を行わない。

圧縮を行う場合は、splSbx/splSpx 構造体の block_size（圧縮ブロックサイズ）を設定しなければなりません。

（4）SplBool ext_rec_flag　（入力）

SBX/SPX ファイルの拡張レコードを書き込むか指定します。

SplTrue: 拡張レコードを書き込む。

SplFalse: 拡張レコードを書き込まない。

（5）splSbx/splSpx* obj　（入力）

ファイルのヘッダー情報、書込データポインタを設定します。

（6）int writeRank　（入力）

書込モード＝0 の場合は設定の必要はありませんが、書込モード＝1 の場合は書き込みを行うノード識別子を設定しなければなりません。

ノード識別子は、書込グループ番号内のローカルノード番号です。

（7）SPL_Comm grp　（入力）

書込モード＝0 の場合は設定の必要はありませんが、書込モード＝1 の場合は書き込みを行うコミュニケーショングループ番号を設定する必要があります。

（使用例）

```
splSbx sbx;                                                    (a)
（省略：初期化、サイズ等の設定）
（省略：書込データポインタ設定）
int mode = 1;
```

```
int writeRank = 5;
SPL_Comm grp = SPL_COMM_WORLD;
if (splWriteSbx("sample.sbx", mode, SplTrue, SplFalse,
                    &sbx, writeRank, grp) != SplTrue) {                    (b)
    return SplFalse;
}
```

(a) splSbx 構造体(ファイル書込情報)を定義、設定します。

(c) splWriteSbx/splWriteSpx 関数を呼び出して書き込みを実行します

４．３　ファイル書込情報破棄（splSbx/splSpx 構造体）

ファイル書込情報（splSbx/splSpx 構造体）使用後、破棄処理を行わなければなりません。破棄の為に" splFinalizeSbx/ splFinalizeSpx"関数を呼び出してください。

関数名	splFinalizeSbx	splSbx 構造体の破棄を行う。。
	splFinalizeSpx	splSpx 構造体の破棄を行う。
引数	splSbx* obj	破棄を行う splSbx 構造体 (splFinalizeSbx の場合)
	splSpx* obj	破棄を行う splSpx 構造体 (splFinalizeSpx の場合)
	SplBool mem_free_flag	メモリ開放フラグ SplTrue:メモリ開放を行う SplFalse:メモリ開放を行わない。
戻り値	SplBool	成功：SplTrue、失敗：SplFalse

メモリ開放フラグは、splSbx/splSpx 構造体の data メンバーに設定したデータポインタを開放するか、しないかのフラグです。

５．SBX/SPX ファイルからの書込パラメータ

SBX/SPX ファイルからの書き込みにおいて書き込み領域の位置、サイズを決定するパラメータとして以下があります。

No	構造体	メンバー	説明
3		size	データサイズ
		wsize	グローバルサイズ
4		start_idx	書込始点インデックス

以下、上記パラメータの設定により読込データの位置、サイズがどのように変更されるか

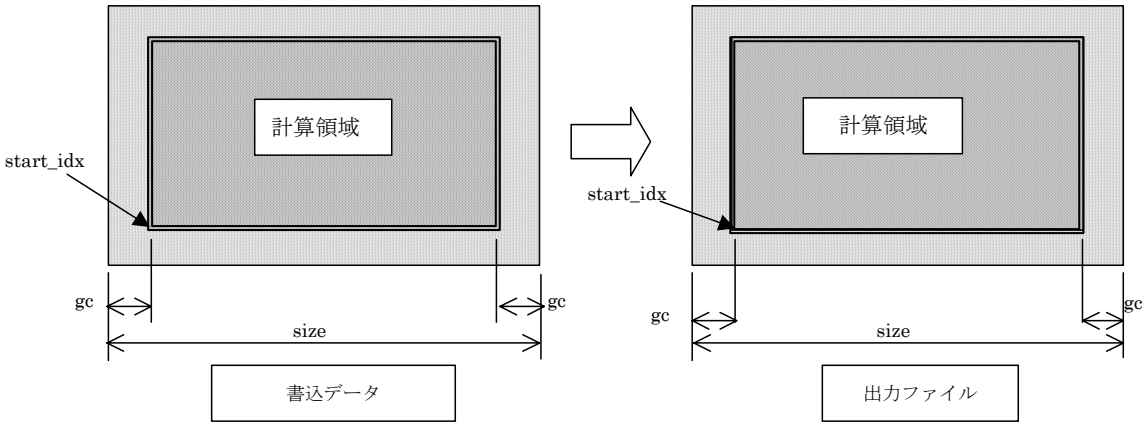
記述します。

５．１　書込サイズの設定

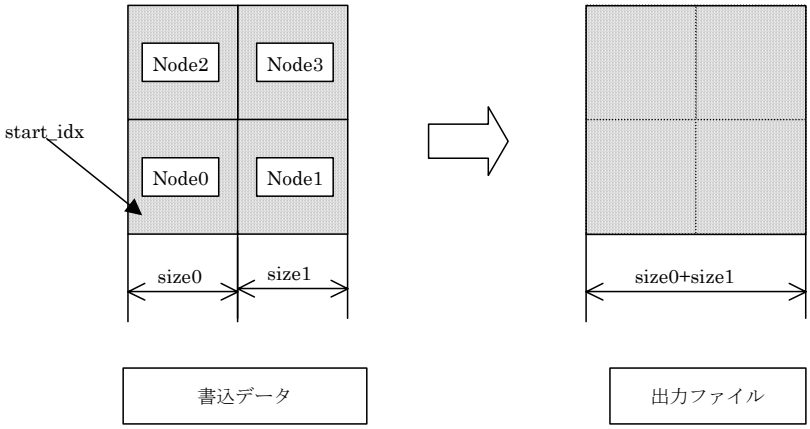
SBX/SPX ファイルに出力するデータサイズは、ファイル書込情報(splSbx/splSpx 構造体)の wsize（グローバルサイズ）の設定によって異なります。

５．１．１　wsize（グローバルサイズ）が設定されていない場合

（単独実行、モード０の場合）



（並列実行・モード１）の場合

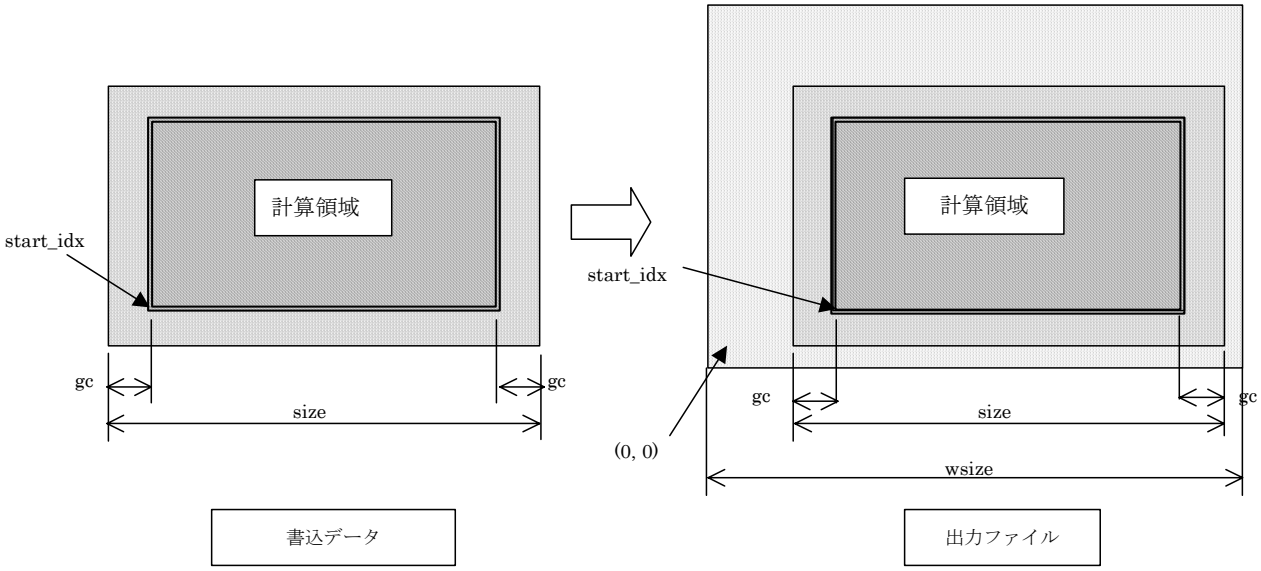


wsize（グローバルサイズ）が０の場合は、始点インデックスから書込データの size 分の領域が出力されます。

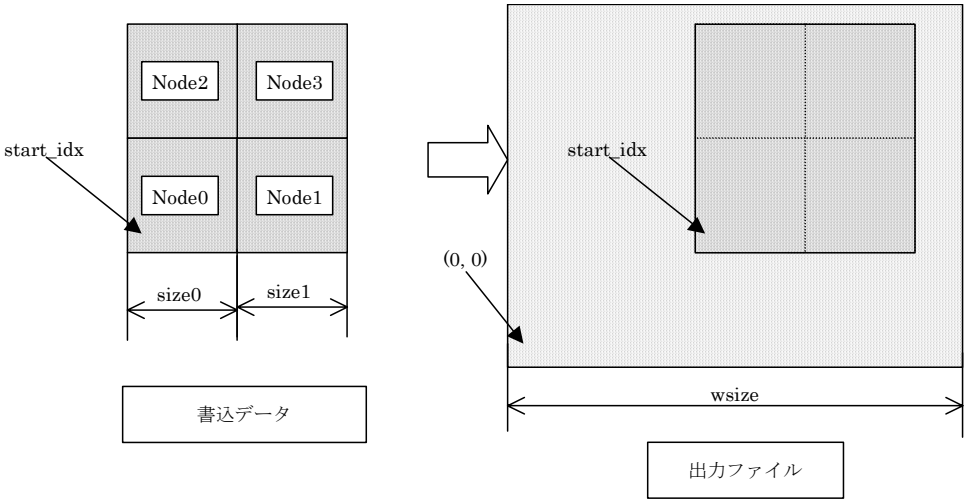
モード１にて、複数ノードからデータ収集して出力する場合は、すべてのノードサイズを包括したサイズにて出力されます。

５．１．１　wsize（グローバルサイズ）が設定されている場合

（単独実行、モード０の場合）



（並列実行・モード１）の場合



wsize（グローバルサイズ）が 0 より大きい場合は、原点(0, 0)から wsize 分の領域が出力されます。

よって、ファイル書込情報(splSbx/splSpx 構造体)のグローバルサイズ、始点インデックスによっては書込データの一部しか出力されないことがあります。

６．SBX/SPX ファイルのヘッダー情報の取得

SBX/SPX ファイルからのヘッダー情報のみ取得は、splGetSbxInfo/ splGetSpxInfo 関数を使用します。

（SBX ファイル）

関数名	splGetSbxInfo	SBX ファイルからヘッダー情報の取得を行う。
引数	const char* fname　[in]	読み込むファイル名
	splSbx* obj　[out]	ヘッダー情報の splSbx 構造体
	int mode　[in]	読込モード
	int readRank　[in]	ノード識別子
	SPL_Comm grp　[in]	コミュニケーショングループ
戻り値	SplBool	成功：SplTrue、失敗：SplFalse

（SPX ファイル）

関数名	splGetSpxInfo	SPX ファイルからヘッダー情報の取得を行う。
引数	const char* fname　[in]	読み込むファイル名
	splSpx* obj　[out]	読込情報の splSpx 構造体
	int mode　[in]	読込モード
	int readRank　[in]	ノード識別子
	SPL_Comm grp　[in]	コミュニケーショングループ
戻り値	SplBool	成功：SplTrue、失敗：SplFalse

以下、引数について説明します。

（１）const char* fname　（入力）

ヘッダー情報を取得するファイル名を設定します。

（２）splSbx/splSpx* obj　（出力）

ファイルのヘッダー情報が返されます。

（３）int mode　（入力）

ファイルの読込モードを設定します。

モードによりファイルをオープンするノード、動作が異なります。

モード番号	説明
0	全てのノードがファイルをオープンして、ファイルを読み込みます。
1	readRank のノードがファイルをオープンし、読み込んだ値を他のノードにブロードキャストします。

（４）int readRank　（入力）

読込モード＝０の場合は設定の必要はありませんが、読込モード＝１の場合は読み込みを行うノード識別子を設定しなければなりません。

ノード識別子は、読込グループ番号内のローカルノード番号です。

（５）SPL_Comm grp　（入力）

読込モード＝０の場合は設定の必要はありませんが、読込モード＝１の場合は読み込みを行うコミュニケーショングループ番号を設定する必要があります。

（使用例）

```
splSbx   sbx;                                                    (a)
// 初期化
if (splInitSbx(&sbx) != SplTrue) return SplFalse;                (b)
int mode = 1;
int readRank = 5;
SPL_Comm grp = SPL_COMM_WORLD;
if (splGetSbxInfo ("sample.sbx", &sbx, mode, readRank, grp) != SplTrue) {    (c)
      return SplFalse;
}
```

(a) splSbx 構造体(ファイル読込情報)を定義します。

(b) splSbx 構造体(ファイル読込情報)を初期化します。

(c) splGetSbxInfo / splGetSpxInfo 関数を呼び出してヘッダー情報の読み込みを実行します

７．読込・書込モード

並列実行時の読み込み、書き込みを行うノードの指定、読込・書込方法によって以下のモードが用意されています。

（読込モード）

モード番号	説明	注意点
0	全てのノードがファイルをオープンして、ファイルを読み込みます。	全てのノードがファイルにアクセスできる環境でなければなりません。
1	readRank のノードがファイルをオープンし、読み込んだ値を他のノードにブロードキャストする。	readRank 以外のノードはファイルリストを作成する必要はありません。 作成しても破棄されます。

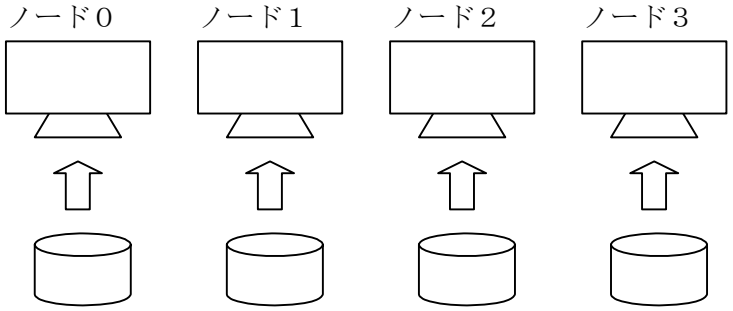
（書込モード）

モード番号	説明	注意点
0	全てのノードがファイルをオープンして、ファイルを書き込みます。	全てのノードがそれぞれの個別のファイルに出力します。
1	writeRank のノードがファイルをオープンし、他のノードから情報を集めてデータを書き出す。	

７．１　読込モード０（モード番号＝０）

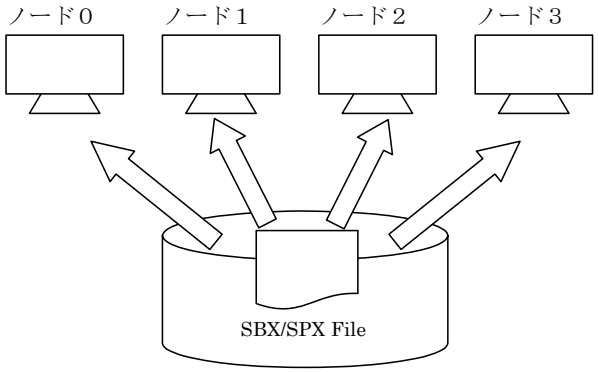
全てのノードがファイルをオープンして、ファイルを読み込みます。

オープンするファイル、読込情報は、すべてのノードにて設定します。



すべてのノードが、ローカルディスクの SBX/SPX ファイルにアクセスできなければなりません。

また共有ディクスを使用する場合は、共有領域に１つ、又はノード別のファイルから読込みを行います。

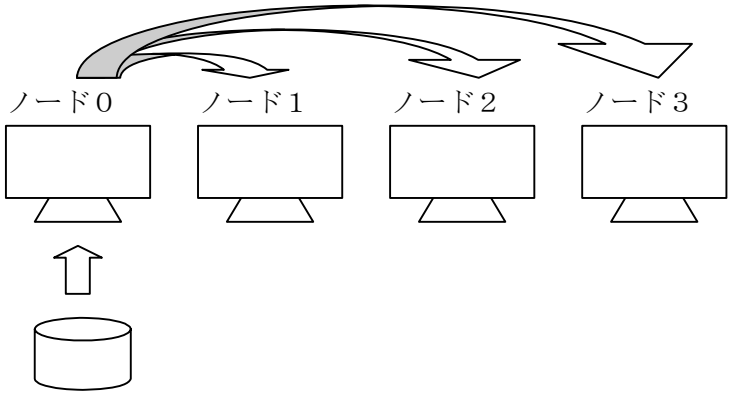


７．２　読込モード１（モード番号＝１）

readRank のノードがファイルをオープンし、読み込んだ値を他のノードにブロードキャストします。

オープンするファイルは readRank のみ設定しますが、各ノードは、自ノードが取り込むデータ領域を設定します。

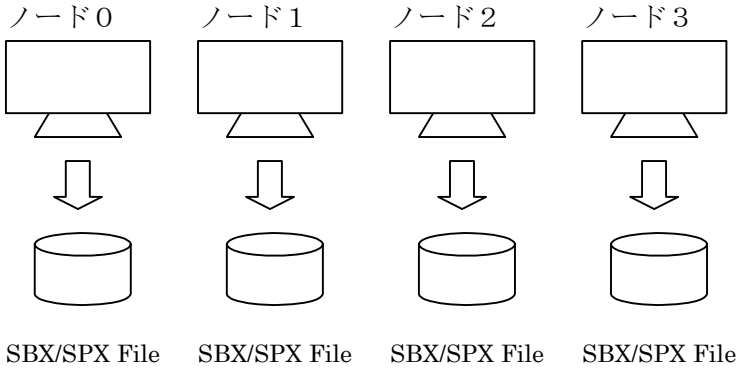
読込ランク番号、読込グループ番号を設定しなければなりません。



７．３　書込モード０（モード番号＝０）

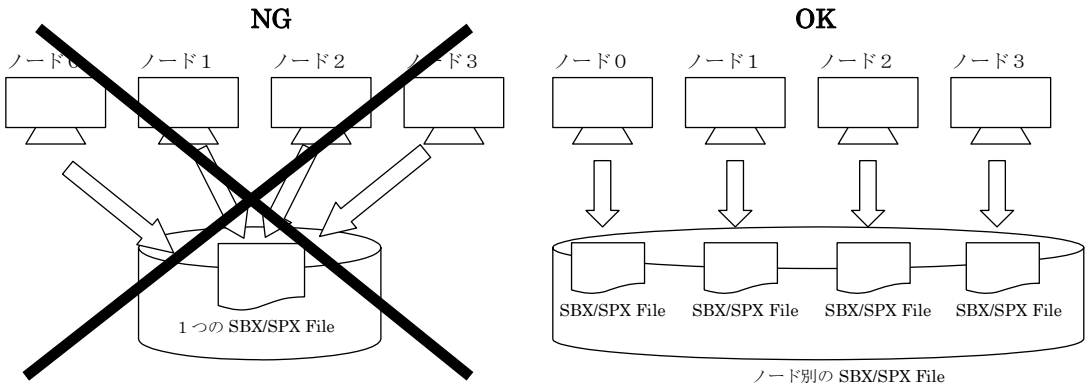
全てのノードがファイルをオープンして、ファイルを書き込みます。

オープンするファイル、書込情報は、すべてのノードにて設定します。



すべてのノードが、ローカルディスクの SBX/SPX ファイルを出力します。

また共有ディクスを使用する場合は、共有領域にノード別のファイルを出力しなければなりません。出力ファイルが１つになるようなファイル指定はできません。

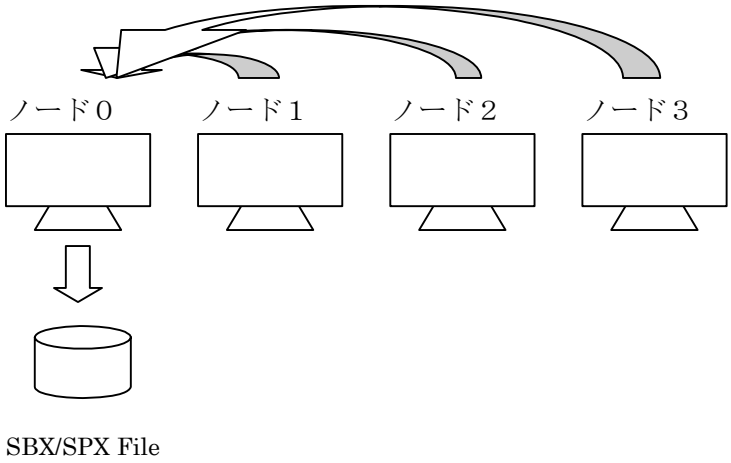


７．４　書込モード１（モード番号＝１）

writeRank のノードがファイルをオープンし、他のノードから情報を集めてデータを書き出しします。

オープンするファイルは writeRank のみ設定しますが、各ノードは自ノードの出力データ領域を設定します。

書込ランク番号、書込グループ番号を設定しなければなりません。



８．サンプルコード

SBX, SPX ファイルからのファイルの読込み、書込みのサンプルコードを以下に記述します。

８．１　ヘッダ情報の取得

SBX, SPX ファイルのヘッダ情報のみを取得します。データは取得しません。

splGetSbxInfo（SBX ファイルヘッダ情報取得）

```
#include <stdio.h>
#include <stdlib.h>
#include "spl/splfio.h"

int main(int argc, char** argv) {
  if (argc <= 1) return EXIT_FAILURE;
  const char* filename = argv[1];                    // SBX ファイル
  printf("file name¥t¥t¥t: %s¥n", filename);

  // splSbx 構造体オブジェクトの生成
  splSbx obj;
  splInitSbx(&obj);

  int mode = 0;
  int rankid = 0;
  SPL_Comm procGrp = SPL_COMM_WORLD;
  // SBX ファイルヘッダ情報を取得する。
  if (!splGetSbxInfo(filename, &obj, mode, rankid, procGrp)) {
    printf("[error_code]=%d,          [error_message]=%s¥n",          spl_get_errcode(),
spl_get_errinfo());
    return EXIT_FAILURE;
  }

  // SBX ファイルヘッダ情報
  printf("dimension¥t¥t¥t: %d¥n", obj.dims);
  printf("vector length¥t¥t¥t: %d¥n", obj.vlen);
  printf("data type¥t¥t¥t: %d¥n", obj.dtype);
```

```
    printf("guide cell size¥t¥t¥t: %ld¥n", obj.gc);
    printf("real data type¥t¥t¥t: %d¥n", obj.rlen);
    printf("origin definision¥t¥t: %d¥n", obj.crddef);
    printf("aux¥t¥t¥t¥t: 0x%02X¥n", obj.aux);
    printf("size[i,j,k]¥t¥t¥t: %ld, %ld, %ld¥n", obj.size[0], obj.size[1], obj.size[2]);
    printf("origin[i,j,k]¥t¥t¥t: %f, %f, %f¥n", obj.origin[0], obj.origin[1], obj.origin[2]);
    printf("pitch[i,j,k]¥t¥t¥t: %f, %f, %f¥n", obj.pitch[0], obj.pitch[1], obj.pitch[2]);
    printf("block size¥t¥t¥t: %ld¥n", obj.block_size);

    // splSbx 構造体オブジェクトの破棄
    splFinalizeSbx(&obj, SplTrue);

    return EXIT_SUCCESS;
}
```

（出力例）

```
file name                    : test_int_comp647.sbx
dimension                    : 3
vector length                : 1
data type                    : 4
guide cell size              : 1
real data type               : 8
origin definision            : 4
aux                          : 0x02
size[i,j,k]                  : 6, 4, 7
origin[i,j,k]                : -90.000000, -175.002000, -60.567000
pitch[i,j,k]                 : 1.000000, 2.000000, 3.000000
block size                   : 20
```

splGetSpxInfo（SPX ファイルヘッダ情報取得）

```
#include <stdio.h>
#include <stdlib.h>
#include "spl/splfio.h"

int main(int argc, char** argv) {
    if (argc <= 1) return EXIT_FAILURE;
```

```
    const char* filename = argv[1];                    // SPX ファイル

    // splSpx 構造体オブジェクトの生成
    splSpx obj;
    splInitSpx(&obj);

    // SPX ファイルヘッダ情報を取得する。
    int mode = 0;
    int rankid = 0;
    SPL_Comm procGrp = SPL_COMM_WORLD;
    if (!splGetSpxInfo(filename, &obj, mode, rankid, procGrp)) {
        printf("[error_code]=%d,        [error_message]=%s¥n",        spl_get_errcode(),
spl_get_errinfo());
        return EXIT_FAILURE;
    }

    // SPX ファイルヘッダ情報
    printf("dimension¥t¥t¥t: %d¥n", obj.dims);
    printf("vector length¥t¥t¥t: %d¥n", obj.vlen);
    printf("data type¥t¥t¥t: %d¥n", obj.dtype);
    printf("guide cell size¥t¥t¥t: %ld¥n", obj.gc);
    printf("real data type¥t¥t¥t: %d¥n", obj.rlen);
    printf("origin definision¥t¥t: %d¥n", obj.crddef);
    printf("aux¥t¥t¥t¥t: 0x%02X¥n", obj.aux);
    printf("size[i,j,k]¥t¥t¥t: %ld, %ld, %ld¥n", obj.size[0], obj.size[1], obj.size[2]);
    printf("origin[i,j,k]¥t¥t¥t: %f, %f, %f¥n", obj.origin[0], obj.origin[1], obj.origin[2]);
    printf("pitch[i,j,k]¥t¥t¥t: %f, %f, %f¥n", obj.pitch[0], obj.pitch[1], obj.pitch[2]);
    printf("step¥t¥t¥t¥t: %ld¥n", obj.step);
    printf("time¥t¥t¥t¥t: %f¥n", obj.time);
    printf("block size¥t¥t¥t: %ld¥n", obj.block_size);

    // splSpx 構造体オブジェクトの破棄
    splFinalizeSpx(&obj, SplTrue);

    return EXIT_SUCCESS;
}
```

（出力例）

```
file name                 : real8_647_00.spx
dimension                 : 3
vector length             : 1
data type                 : 1
guide cell size           : 1
real data type            : 8
origin definision         : 2
aux                       : 0x04
size[i,j,k]               : 6, 4, 7
origin[i,j,k]             : -90.000000, -175.002000, -60.567000
pitch[i,j,k]              : 1.000000, 2.000000, 3.000000
step                      : 150
time                      : 0.150000
block size                : 24
```

８．２　ファイルの読込み

SBX, SPX ファイルからデータの読込みを行います。

splReadSbx（SBX ファイルの読込み）

```
#include <stdio.h>
#include <stdlib.h>
#include "spl/splfio.h"

int main(int argc, char** argv) {
  if (argc <= 1) return EXIT_FAILURE;
  const char* filename = argv[1];                    // SBX ファイル
  printf("file name¥t¥t¥t: %s¥n", filename);

  // splSbx 構造体オブジェクトの生成
  splSbx obj;
  splInitSbx(&obj);

  // SBX ファイル読込情報の設定を行う。
  obj.dims = 3;                                      // 3 次元
  obj.vlen = 1;                                      // 1 固定
  obj.dtype = 1;                                     // データ型=char
  obj.gc = 1;                                        // ガイドセル数
  obj.start_idx[0] = obj.start_idx[1] = obj.start_idx[2] = 0;    // 始点インデックス
  obj.size[0] = 6; obj.size[1] = 4; obj.size[2] = 3;             // 読込サイズ={6,4,3}
  // 読込データ領域の確保、初期化
  obj.data.c = (unsigned char*)malloc(obj.size[0]*obj.size[1]*obj.size[2]*obj.dtype);
  memset(obj.data.c, 0x00, obj.size[0]*obj.size[1]*obj.size[2]*obj.dtype);

  // 読込 SBX ファイル名をファイルリストに設定を行う。
  splSvFileList file_list;
  splInitSvFileList(&file_list);
  splAddSvFile(&file_list, filename, obj.start_idx);

  // SBX ファイルの読込み
  int mode = 0;
```

```
  int rankid = 0;
  SPL_Comm procGrp = SPL_COMM_WORLD;
  if (!splReadSbx(&file_list, mode, &obj, rankid, procGrp)) {
    return EXIT_FAILURE;
  }

  // 読込データの出力
  int j, k;
  for (k=0; k<obj.size[2]; k++) {
    for (j=0; j<obj.size[1]; j++) {
      int idx = k*obj.size[0]*obj.size[1]+j*obj.size[0];
      printf("k=%d,j=%d [%02X %02X %02X %02X %02X %02X]¥n",
k, j,
obj.data.c[idx], obj.data.c[idx+1],obj.data.c[idx+2],
obj.data.c[idx+3],obj.data.c[idx+4],obj.data.c[idx+5]);
    }
    printf("¥n");
  }

  // ファイルリストの破棄
  splFinalizeSvFileList(&file_list);

  // splSbx 構造体オブジェクトの破棄
  splFinalizeSbx(&obj, SplTrue);

  return EXIT_SUCCESS;
}
```

（出力例）

```
file name                          : test_char_uncomp_643.sbx
k=0,j=0 [00 01 02 03 04 05]
k=0,j=1 [10 11 12 13 14 15]
k=0,j=2 [20 21 22 23 24 25]
k=0,j=3 [30 31 32 33 34 35]

k=1,j=0 [00 01 02 03 04 05]
k=1,j=1 [10 11 12 13 14 15]
```

```
k=1,j=2 [20 21 22 23 24 25]
k=1,j=3 [30 31 32 33 34 35]

k=2,j=0 [00 01 02 03 04 05]
k=2,j=1 [10 11 12 13 14 15]
k=2,j=2 [20 21 22 23 24 25]
k=2,j=3 [30 31 32 33 34 35]
```

splReadSpx（SPX ファイルの読込み）

```
#include <stdio.h>
#include <stdlib.h>
#include "spl/splfio.h"

int main(int argc, char** argv) {
  if (argc <= 1) return EXIT_FAILURE;
  const char* filename = argv[1];                    // SPX ファイル
  printf("file name¥t¥t¥t: %s¥n", filename);

  // splSpx 構造体オブジェクトの生成
  splSpx obj;
  splInitSpx(&obj);

  // SPX ファイル読込情報の設定を行う。
  obj.dims = 3;                                      // 3 次元
  obj.vlen = 1;                                      // ベクトル長
  obj.dtype = 1;                                     // データタイプ
  obj.gc = 1;                                        // ガイドセル数
  obj.rlen = 4;                                      // 実数データサイズ(=float)
  obj.start_idx[0] = obj.start_idx[1] = obj.start_idx[2] = 0;    // 始点インデックス
  obj.size[0] = 3; obj.size[1] = 4; obj.size[2] = 3;             // 読込サイズ={3,4,3}
  // 読込データ領域の確保、初期化
  obj.data.c = (unsigned
char*)malloc(obj.size[0]*obj.size[1]*obj.size[2]*obj.vlen*obj.rlen);
  memset(obj.data.c, 0x00, obj.size[0]*obj.size[1]*obj.size[2]*obj.vlen*obj.rlen);
```

```
  // 読込 SPX ファイル名をファイルリストに設定を行う。
  splSvFileList file_list;
  splInitSvFileList(&file_list);
  splAddSvFile(&file_list, filename, obj.start_idx);

  // SPX ファイルの読込み
  int mode = 0;
  int rankid = 0;
  SPL_Comm procGrp = SPL_COMM_WORLD;
  if (!splReadSpx(&file_list, mode, &obj, rankid, procGrp)) {
    return EXIT_FAILURE;
  }

  // 読込データの出力
  int j, k;
  for (k=0; k<obj.size[2]; k++) {
    for (j=0; j<obj.size[1]; j++) {
      int idx = (k*obj.size[0]*obj.size[1]+j*obj.size[0])*obj.vlen*obj.rlen;
      printf("k=%d,j=%d [", k, j];
      printf("%02X%02X%02X%02X",
 obj.data.c[idx], obj.data.c[idx+1],obj.data.c[idx+2],obj.data.c[idx+3]);
      idx += obj.vlen*obj.rlen;
      printf(" %02X%02X%02X%02X",
obj.data.c[idx], obj.data.c[idx+1],obj.data.c[idx+2],obj.data.c[idx+3]);
      idx += obj.vlen*obj.rlen;
      printf(" %02X%02X%02X%02X",
obj.data.c[idx], obj.data.c[idx+1],obj.data.c[idx+2],obj.data.c[idx+3]);
      printf("]¥n");
    }
    printf("¥n");
  }
  // ファイルリストの破棄
  splFinalizeSvFileList(&file_list);

  // splSpx 構造体オブジェクトの破棄
  splFinalizeSpx(&obj, SplTrue);
```

```
   return EXIT_SUCCESS;
}
```

（出力例）

```
file name                              : float_test.spx
k=0,j=0 [00000000 01000000 02000000]
k=0,j=1 [08000000 09000000 0A000000]
k=0,j=2 [10000000 11000000 12000000]
k=0,j=3 [18000000 19000000 1A000000]

k=1,j=0 [40000000 41000000 42000000]
k=1,j=1 [48000000 49000000 4A000000]
k=1,j=2 [50000000 51000000 52000000]
k=1,j=3 [58000000 59000000 5A000000]

k=2,j=0 [80000000 81000000 82000000]
k=2,j=1 [88000000 89000000 8A000000]
k=2,j=2 [90000000 91000000 92000000]
k=2,j=3 [98000000 99000000 9A000000]
```

８．３　ファイルの書込み

SBX, SPX ファイルにデータの書込みを行います。

splWriteSbx（SBX ファイルの書込み）

```
#include <stdio.h>
#include <stdlib.h>
#include "spl/splfio.h"

int main(int argc, char** argv) {
  if (argc <= 1) return EXIT_FAILURE;
  const char* filename = argv[1];                    // SBX 書込ファイル
  printf("file name¥t¥t¥t: %s¥n", filename);

  // splSbx 構造体オブジェクトの生成
  splSbx obj;
  splInitSbx(&obj);

  // SBX ファイル書込情報の設定を行う。
  obj.dims = 3;                                      // 3 次元
  obj.vlen = 1;                                      // 1 固定
  obj.dtype = 1;                                     // データ型=char
  obj.gc = 1;                                        // ガイドセル数
  obj.rlen = 4;                                      // 実数サイズ
  obj.crddef = 1;                                    // 座標定義位置
  obj.aux = 0x02;                                    // AUX
  obj.size[0] = 6; obj.size[1] = 4; obj.size[2] = 3;          // 読込サイズ={6,4,3}
  obj.origin[0] = 10.0; obj.origin[1] = 20.0; obj.origin[2] = 15.0;   // 原点座標
  obj.pitch[0] = obj.pitch[1] = obj.pitch[2] = 1.0;   // ピッチ
  obj.start_idx[0] = obj.start_idx[1] = obj.start_idx[2] = 0;   // 始点インデックス
  obj.block_size = obj.size[0]*obj.size[1];      // 圧縮ブロックサイズ

  // 書込データ（ダミーデータ作成）
  obj.data.c = (unsigned char*)malloc(obj.size[0]*obj.size[1]*obj.size[2]*obj.dtype);
  int i;
  for (i=0; i<obj.size[0]*obj.size[1]*obj.size[2]; i++) obj.data.c[i] = i;
```

```
  // SPX ファイルの書込み
  int mode = 0;
  SplBool cmp_flag = SplTrue;                // 圧縮データ書込
  SplBool ext_rec_flag = SplFalse;              // 拡張レコード書込なし
  int rankid = 0;
  SPL_Comm procGrp = SPL_COMM_WORLD;
  if (!splWriteSbx(filename, mode,
                        cmp_flag, ext_rec_flag, obj.block_size,
                        &obj, rankid, procGrp)) {
    return EXIT_FAILURE;
  }

  // splSbx 構造体オブジェクトの破棄
  splFinalizeSbx(&obj, SplTrue);            //書込データの破棄を行わない場合は、
SphFalse

  return EXIT_SUCCESS;
}
```

splWriteSpx（SPX ファイルの書込み）

```
#include <stdio.h>
#include <stdlib.h>
#include "spl/splfio.h"

int main(int argc, char** argv) {
  if (argc <= 1) return EXIT_FAILURE;
  const char* filename = argv[1];                    // SPX 書込ファイル
  printf("file name¥t¥t¥t: %s¥n", filename);

  // splSpx 構造体オブジェクトの生成
  splSpx obj;
  splInitSpx(&obj);

  // SPX ファイル書込情報の設定を行う。
```

```
  obj.dims = 3;                                  // 3 次元
  obj.vlen = 1;                                  // 1 固定
  obj.dtype = 1;                                 // データタイプ＝スカラデータ
  obj.gc = 1;                                    // ガイドセル数
  obj.rlen = 8;                                  // 実数データサイズ(=double)
  obj.crddef = 1;                                // 座標定義位置
  obj.aux = 0x02;                                // AUX
  obj.size[0] = 6; obj.size[1] = 4; obj.size[2] = 3;          // 読込サイズ={6,4,3}
  obj.origin[0] = 10.0; obj.origin[1] = 20.0; obj.origin[2] = 15.0;    // 原点座標
  obj.pitch[0] = obj.pitch[1] = obj.pitch[2] = 1.0;    // ピッチ
  obj.start_idx[0] = obj.start_idx[1] = obj.start_idx[2] = 0;    // 始点インデックス
  obj.block_size = obj.size[0]*obj.size[1];       // 圧縮ブロックサイズ


  // 書込データ（ダミーデータ作成）
  obj.data.c                                     =                                    (unsigned
char*)malloc(obj.size[0]*obj.size[1]*obj.size[2]*obj.rlen*obj.vlen);
  int i;
  double value = 1.0;double step = 0.1;
  for (i=0; i<obj.size[0]*obj.size[1]*obj.size[2]; i++) obj.data.d[i] = value+step*i;

  // SPX ファイルの書込み
  int mode = 0;
  SplBool cmp_flag = SplTrue;                // 圧縮データ書込
  SplBool ext_rec_flag = SplFalse;               // 拡張レコード書込なし
  int rankid = 0;
  SPL_Comm procGrp = SPL_COMM_WORLD;
  if (!splWriteSpx(filename, mode,
                   cmp_flag, ext_rec_flag, obj.block_size,
                   &obj, rankid, procGrp)) {
    return EXIT_FAILURE;
  }

  // splSpx 構造体オブジェクトの破棄
  splFinalizeSpx(&obj, SplTrue);             //書込データの破棄を行わない場合は、
SphFalse
```

```
    return EXIT_SUCCESS;
}
```