Reading From or Writing To a Subset of a Dataset

There are two ways that you can select a subset in an HDF5 dataset and read or write to it:

> ***Hyperslab Selection*** : The `H5S_SELECT_HYPERSLAB` call selects a logically contiguous collection of points in a dataspace, or a regular pattern of points or blocks in a dataspace.
>
> ***Element Selection****:* The `H5S_SELECT_ELEMENTS` call selects elements in an array.

HDF5 allows you to read from or write to a portion or subset of a dataset by:

- Selecting a Subset of the Dataset's Dataspace,
- Selecting a Memory Dataspace,
- Reading From or Writing to a Dataset Subset.

Selecting a Subset of the Dataset's Dataspace

First you must obtain the dataspace of a dataset in a file by calling H5D_GET_SPACE .

Then select a subset of that dataspace by calling H5S_SELECT_HYPERSLAB. The *offset*, *count*, *stride* and *block* parameters of this API define the shape and size of the selection. They must be arrays with the same number of dimensions as the rank of the dataset's dataspace. These arrays **ALL** work together to define a selection. A change to one of these arrays can affect the others.

> *offset*: An array that specifies the offset of the starting element of the specified hyperslab.
>
> *count*: An array that determines how many blocks to select from the dataspace in each dimension. If the block size for a dimension is one then the *count* is the number of elements along that dimension.
>
> *stride*: An array that allows you to sample elements along a dimension. For example, a stride of one (or NULL) will select every element along a dimension, a stride of two will select every other element, and a stride of three will select an element after every two elements.
>
> *block*: An array that determines the size of the element block selected from a dataspace. If the block size is one or NULL then the block size is a single element in that dimension.

Selecting a Memory Dataspace

You must select a memory dataspace in addition to a file dataspace before you can read a subset from or write a subset to a dataset. A memory dataspace can be specified by calling H5S_CREATE_SIMPLE.

The memory dataspace passed to the read or write call *must* contain the same number of elements as the file dataspace. The number of elements in a dataspace selection can be determined with the H5S_GET_SELECT_NPOINTS API.

Reading From or Writing To a Dataset Subset

To read from or write to a dataset subset, the H5D_READ and H5D_WRITE routines are used. The memory and file dataspace identifiers from the selections that were made are passed into the read or write call. For example (C):

```
status = H5Dwrite (.., .., memspace_id, dataspace_id, .., ..);
```

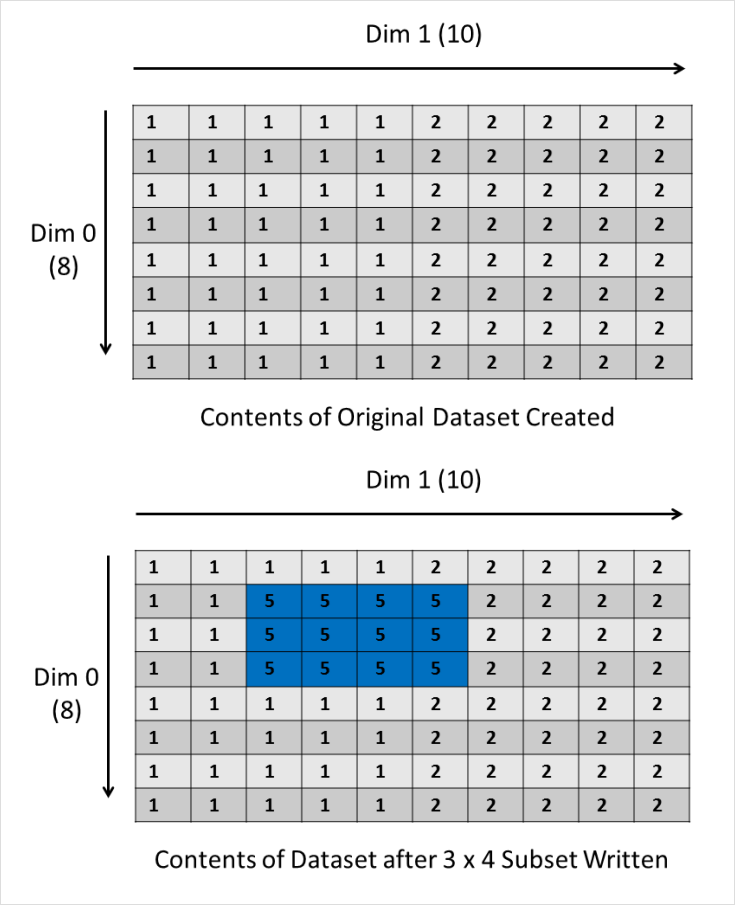
Programming Example

Description

This example creates an 8 x 10 integer dataset in an HDF5 file. It then selects and writes to a 3 x 4 subset of the dataset created with the dimensions offset by 1 x 2. (If using Fortran, the dimensions will be swapped. The dataset will be 10 x 8, the subset will be 4 x 3, and the offset will be 2 x 1.)

PLEASE NOTE that the examples and images below were created using C.

The following image shows the dataset that gets written originally, and the subset of data that gets modified afterwards. Dimension 0 is vertical and Dimension 1 is horizontal as shown below:



The subset on the right above is created using these values for offset, count stride, and block:

```
offset = {1, 2}
count = {3, 4}
stride = {1, 1}
block = {1, 1}
```

Sample code:

C

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 * Copyright by The HDF Group.                                               *
 * Copyright by the Board of Trustees of the University of Illinois.         *
 * All rights reserved.                                                      *
 *                                                                           *
 * This file is part of HDF5.  The full HDF5 copyright notice, including     *
 * terms governing use, modification, and redistribution, is contained in    *
 * the files COPYING and Copyright.html.  COPYING can be found at the root   *
 * of the source code distribution tree; Copyright.html can be found at the  *
 * root level of an installed copy of the electronic HDF5 document set and   *
 * is linked from the top-level documents page.  It can also be found at     *
 * http://hdfgroup.org/HDF5/doc/Copyright.html.  If you do not have          *
 * access to either file, you may request a copy from help@hdfgroup.org.     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/*
 *  This example illustrates how to read/write a subset of data (a slab)
 *  from/to a dataset in an HDF5 file.  It is used in the HDF5 Tutorial.
```

```
 */

#include "hdf5.h"

#define FILE        "subset.h5"
#define DATASETNAME "IntArray"
#define RANK  2

#define DIM0_SUB  3                         /* subset dimensions */
#define DIM1_SUB  4


#define DIM0     8                          /* size of dataset */
#define DIM1     10

int
main (void)
{
    hsize_t     dims[2], dimsm[2];
    int         data[DIM0][DIM1];           /* data to write */
    int         sdata[DIM0_SUB][DIM1_SUB];  /* subset to write */
    int         rdata[DIM0][DIM1];          /* buffer for read */

    hid_t       file_id, dataset_id;        /* handles */
    hid_t       dataspace_id, memspace_id;

    herr_t      status;

    hsize_t     count[2];              /* size of subset in the file */
    hsize_t     offset[2];             /* subset offset in the file */
    hsize_t     stride[2];
    hsize_t     block[2];
    int         i, j;


    /*************************************************************
     * Create a new file with default creation and access properties.*
     * Then create a dataset and write data to it and close the file *
     * and dataset.                                                  *
     *************************************************************/

    file_id = H5Fcreate (FILE, H5F_ACC_TRUNC, H5P_DEFAULT, H5P_DEFAULT);

    dims[0] = DIM0;
    dims[1] = DIM1;
    dataspace_id = H5Screate_simple (RANK, dims, NULL);

    dataset_id = H5Dcreate2 (file_id, DATASETNAME, H5T_STD_I32BE, dataspace_id,
                            H5P_DEFAULT, H5P_DEFAULT, H5P_DEFAULT);


    for (j = 0; j < DIM0; j++) {
 for (i = 0; i < DIM1; i++)
            if (i< (DIM1/2))
        data[j][i] = 1;
            else
               data[j][i] = 2;
    }
```

```
   status = H5Dwrite (dataset_id, H5T_NATIVE_INT, H5S_ALL, H5S_ALL,
                      H5P_DEFAULT, data);

   printf ("\nData Written to File:\n");
   for (i = 0; i<DIM0; i++){
      for (j = 0; j<DIM1; j++)
          printf (" %i", data[i][j]);
      printf ("\n");
   }
   status = H5Sclose (dataspace_id);
   status = H5Dclose (dataset_id);
   status = H5Fclose (file_id);


   /*************************************************
    * Reopen the file and dataset and write a subset of *
    * values to the dataset.
    *************************************************/

   file_id = H5Fopen (FILE, H5F_ACC_RDWR, H5P_DEFAULT);
   dataset_id = H5Dopen2 (file_id, DATASETNAME, H5P_DEFAULT);

   /* Specify size and shape of subset to write. */

   offset[0] = 1;
   offset[1] = 2;

   count[0]  = DIM0_SUB;
   count[1]  = DIM1_SUB;

   stride[0] = 1;
   stride[1] = 1;

   block[0] = 1;
   block[1] = 1;

   /* Create memory space with size of subset. Get file dataspace
      and select subset from file dataspace. */

   dimsm[0] = DIM0_SUB;
   dimsm[1] = DIM1_SUB;
   memspace_id = H5Screate_simple (RANK, dimsm, NULL);

   dataspace_id = H5Dget_space (dataset_id);
   status = H5Sselect_hyperslab (dataspace_id, H5S_SELECT_SET, offset,
                                 stride, count, block);

   /* Write a subset of data to the dataset, then read the
      entire dataset back from the file.  */

   printf ("\nWrite subset to file specifying:\n");
   printf ("    offset=1x2 stride=1x1 count=3x4 block=1x1\n");
   for (j = 0; j < DIM0_SUB; j++) {
for (i = 0; i < DIM1_SUB; i++)
   sdata[j][i] = 5;
   }

   status = H5Dwrite (dataset_id, H5T_NATIVE_INT, memspace_id,
                      dataspace_id, H5P_DEFAULT, sdata);
```

```
status = H5Dread (dataset_id, H5T_NATIVE_INT, H5S_ALL, H5S_ALL,
                  H5P_DEFAULT, rdata);

printf ("\nData in File after Subset is Written:\n");
for (i = 0; i<DIM0; i++){
   for (j = 0; j<DIM1; j++)
       printf (" %i", rdata[i][j]);
   printf ("\n");
}

status = H5Sclose (memspace_id);
status = H5Sclose (dataspace_id);
status = H5Dclose (dataset_id);
status = H5Fclose (file_id);
```

```
}
```

Fortran

```
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
!  Copyright by The HDF Group.                                               *
!  Copyright by the Board of Trustees of the University of Illinois.         *
!  All rights reserved.                                                      *
!                                                                            *
!  This file is part of HDF5.  The full HDF5 copyright notice, including     *
!  terms governing use, modification, and redistribution, is contained in    *
!  the files COPYING and Copyright.html.  COPYING can be found at the root   *
!  of the source code distribution tree; Copyright.html can be found at the  *
!  root level of an installed copy of the electronic HDF5 document set and   *
!  is linked from the top-level documents page.  It can also be found at     *
!  http://hdfgroup.org/HDF5/doc/Copyright.html.  If you do not have          *
!  access to either file, you may request a copy from help@hdfgroup.org.     *
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
!
! This example shows how to write and read a hyperslab.
! It is used in the HDF5 Tutorial.
!

PROGRAM H5_SUBSET

  USE HDF5 ! This module contains all necessary modules

  IMPLICIT NONE

  CHARACTER(LEN=9), PARAMETER :: filename = "subset.h5"  ! File name
  CHARACTER(LEN=8), PARAMETER :: dsetname = "IntArray"   ! Dataset name

  INTEGER(HID_T) :: file_id       ! File identifier
  INTEGER(HID_T) :: dset_id       ! Dataset identifier
  INTEGER(HID_T) :: dataspace     ! Dataspace identifier
  INTEGER(HID_T) :: memspace      ! memspace identifier

  !
  ! To change the subset size, modify size of dimsm, sdata, dim0_sub,
  ! dim1_sub, and count
  !
  INTEGER(HSIZE_T), DIMENSION(1:2) :: dimsm = (/4,3/) ! Dataset dimensions
  INTEGER, DIMENSION(1:4,1:3) :: sdata                ! Subset buffer
  INTEGER :: dim0_sub = 4
  INTEGER :: dim1_sub = 3
  INTEGER(HSIZE_T), DIMENSION(1:2) :: count = (/4,3/)  ! Size of hyperslab
  INTEGER(HSIZE_T), DIMENSION(1:2) :: offset = (/2,1/) ! Hyperslab offset
  INTEGER(HSIZE_T), DIMENSION(1:2) :: stride = (/1,1/) ! Hyperslab stride
  INTEGER(HSIZE_T), DIMENSION(1:2) :: block = (/1,1/)  ! Hyperslab block size

  INTEGER(HSIZE_T), DIMENSION(1:2) :: dimsf = (/10,8/) ! Dataset dimensions


  INTEGER, DIMENSION(1:10,1:8) :: data     ! Data to write
```

```
INTEGER, DIMENSION(1:10,1:8) :: rdata    ! Data to read

INTEGER :: rank = 2      ! Dataset rank ( in file )
INTEGER :: dim0 = 10     ! Dataset size in file
INTEGER :: dim1 = 8

INTEGER :: i, j

INTEGER :: error         ! Error flag
INTEGER(HSIZE_T), DIMENSION(2) :: data_dims

!
! Write data to the HDF5 file.
!

!
! Data initialization.
!
DO i = 1, dim0
   DO j = 1, dim1
      IF (i .LE. (dim0 / 2)) THEN
         data(i,j) = 1
      ELSE
         data(i,j) = 2
      END IF
   END DO
END DO

!
! Initialize FORTRAN interface.
!
CALL h5open_f(error)

!
! Create a new file using default properties.
!
CALL h5fcreate_f(filename, H5F_ACC_TRUNC_F, file_id, error)

!
! Create the data space for the  dataset.
!
CALL h5screate_simple_f(rank, dimsf, dataspace, error)

!
! Create the dataset with default properties.
!
CALL h5dcreate_f(file_id, dsetname, H5T_NATIVE_INTEGER, dataspace, &
     dset_id, error)

!
! Write the dataset.
!
data_dims(1) = dim0
data_dims(2) = dim1
CALL h5dwrite_f(dset_id, H5T_NATIVE_INTEGER, data, data_dims, error)

!
! Data Written to File
!
```

```
WRITE(*,'(/,A)') "Original Data Written to File:"
DO i = 1, dim0
   WRITE(*,'(100(1X,I0,1X))') DATA(i,1:dim1)
END DO

!
!
! Close the dataspace, dataset, and file.
!
CALL h5sclose_f(dataspace, error)
CALL h5dclose_f(dset_id, error)
CALL h5fclose_f(file_id, error)

!
! Initialize subset data array.
!
sdata(1:dim0_sub,1:dim1_sub) = 5

!
! Open the file.
!
CALL h5fopen_f(filename, H5F_ACC_RDWR_F, file_id, error)

!
! Open the  dataset.
!
CALL h5dopen_f(file_id, dsetname, dset_id, error)

!
! Get dataset's dataspace identifier and select subset.
!
CALL h5dget_space_f(dset_id, dataspace, error)
CALL h5sselect_hyperslab_f(dataspace, H5S_SELECT_SET_F, &
     offset, count, error, stride, BLOCK)
!
! Create memory dataspace.
!
CALL h5screate_simple_f(rank, dimsm, memspace, error)

WRITE(*,'(/,A)') "Write subset to file specifying:"
WRITE(*,'(A,/)') "   offset=2x1 stride=1x1 count=4x3 block=1x1"

!
! Write subset to dataset
!
data_dims(1:2) = (/dim0_sub, dim1_sub/)
CALL h5dwrite_f(dset_id, H5T_NATIVE_INTEGER, sdata, data_dims, error, &
     memspace, dataspace)

data_dims(1:2) = (/dim0, dim1/)
CALL h5dread_f(dset_id, H5T_NATIVE_INTEGER, rdata, data_dims, error)

!
! Read entire dataset back
!
WRITE(*,'(A)') "Data in File after Subset Written:"
DO i = 1, dim0
  WRITE(*,'(100(1X,I0,1X))') rdata(i,1:dim1)
END DO
```

```
PRINT *, " "

!
! Close everything opened.
!
CALL h5sclose_f(dataspace, error)
CALL h5sclose_f(memspace, error)
CALL h5dclose_f(dset_id, error)
CALL h5fclose_f(file_id, error)

!
! Close FORTRAN interface.
!
CALL h5close_f(error)
```

```
END PROGRAM H5_SUBSET
```

C++

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 * Copyright by The HDF Group.           *
 * Copyright by the Board of Trustees of the University of Illinois.      *
 * All rights reserved.            *
 *                                                                         *
 * This file is part of HDF5.  The full HDF5 copyright notice, including     *
 * terms governing use, modification, and redistribution, is contained in    *
 * the files COPYING and Copyright.html.  COPYING can be found at the root   *
 * of the source code distribution tree; Copyright.html can be found at the  *
 * root level of an installed copy of the electronic HDF5 document set and   *
 * is linked from the top-level documents page.  It can also be found at     *
 * http://hdfgroup.org/HDF5/doc/Copyright.html.  If you do not have      *
 * access to either file, you may request a copy from help@hdfgroup.org.     *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/*
 *  This example illustrates how to read/write a subset of data (a slab)
 *  from/to a dataset in an HDF5 file. It is used in the HDF5 Tutorial.
 */

#include <iostream>
#include <string>

#include "H5Cpp.h"

#ifndef H5_NO_NAMESPACE
     using namespace H5;
#ifndef H5_NO_STD
     using std::cout;
     using std::endl;
#endif  // H5_NO_STD
#endif

const H5std_string FILE_NAME("h5tutr_subset.h5");
const H5std_string DATASET_NAME("IntArray");

const int     RANK = 2;
const int     DIM0_SUB = 3; // subset dimensions
const int     DIM1_SUB = 4;
const int     DIM0 = 8;  // size of dataset
const int     DIM1 = 10;

int main (void)
{
    int     i,j;
    int     data[DIM0][DIM1], sdata[DIM0_SUB][DIM1_SUB], rdata[DIM0][DIM1];

    // Try block to detect exceptions raised by any of the calls inside it
    try
    {
```

```
// Turn off the auto-printing when failure occurs so that we can
// handle the errors appropriately
Exception::dontPrint();

// -------------------------------------------------
// Create a new file using the default property lists.
// Then create a dataset and write data to it.
// Close the file and dataset.
// -------------------------------------------------

H5File file(FILE_NAME, H5F_ACC_TRUNC);

hsize_t dims[2];
dims[0] = DIM0;
dims[1] = DIM1;
DataSpace dataspace = DataSpace (RANK, dims);

DataSet dataset(file.createDataSet( DATASET_NAME,
                                PredType::STD_I32BE, dataspace) );


for (j = 0; j < DIM0; j++) {
     for (i = 0; i < DIM1; i++)
         if (i< (DIM1/2))
            data[j][i] = 1;
         else
            data[j][i] = 2;
      }

dataset.write(data, PredType::NATIVE_INT);

cout << endl << "Data Written to File:" << endl;
for (j = 0; j < DIM0; j++) {
    for (i = 0; i < DIM1; i++)
       cout << " " <<  data[j][i];
    cout << endl;
}

dataspace.close();
dataset.close();
file.close();

// -------------------------------------------------
// Reopen the file and dataset and write a subset of
// values to the dataset.
// -------------------------------------------------

hsize_t offset[2], count[2], stride[2], block[2];
hsize_t dimsm[2];

file.openFile(FILE_NAME, H5F_ACC_RDWR);
dataset = file.openDataSet(DATASET_NAME);

// Specify size and shape of subset to write.

offset[0] = 1;
offset[1] = 2;

count[0]  = DIM0_SUB;
```

```
count[1]  = DIM1_SUB;

stride[0] = 1;
stride[1] = 1;

block[0] = 1;
block[1] = 1;

// Define Memory Dataspace. Get file dataspace and select
// a subset from the file dataspace.

dimsm[0] = DIM0_SUB;
dimsm[1] = DIM1_SUB;

DataSpace memspace(RANK, dimsm, NULL);

dataspace = dataset.getSpace();
dataspace.selectHyperslab(H5S_SELECT_SET, count, offset, stride, block);

// Write a subset of data to the dataset, then read the
// entire dataset back from the file.

cout << endl << "Write subset to file specifying: " << endl;
cout << "  offset=1x2 stride=1x1 count=3x4 block=1x1" << endl;
for (j = 0; j < DIM0_SUB; j++) {
    for (i = 0; i < DIM1_SUB; i++)
       sdata[j][i] = 5;
}

dataset.write(sdata, PredType::NATIVE_INT, memspace, dataspace);
dataset.read(rdata, PredType::NATIVE_INT);


cout << endl << "Data in File after Subset is Written:" << endl;
for (i = 0; i < DIM0; i++) {
    for (j = 0; j < DIM1; j++)
       cout << " " <<  rdata[i][j];
    cout << endl;
}
cout << endl;

// It is not necessary to close these objects because close() will
// be called when the object instances are going out of scope.
dataspace.close();
memspace.close();
dataset.close();
file.close();

   }  // end of try block

   // catch failure caused by the H5File operations
   catch(FileIException error)
   {
error.printError();
return -1;
   }

   // catch failure caused by the DataSet operations
   catch(DataSetIException error)
```

```
   {
error.printError();
return -1;
   }

   // catch failure caused by the DataSpace operations
   catch(DataSpaceIException error)
   {
error.printError();
return -1;
   }

   return 0;  // successfully terminated
```

```
}
```

See HDF5 Introductory Examples for the examples used in the Learning the Basics tutorial. There are examples for several other languages.

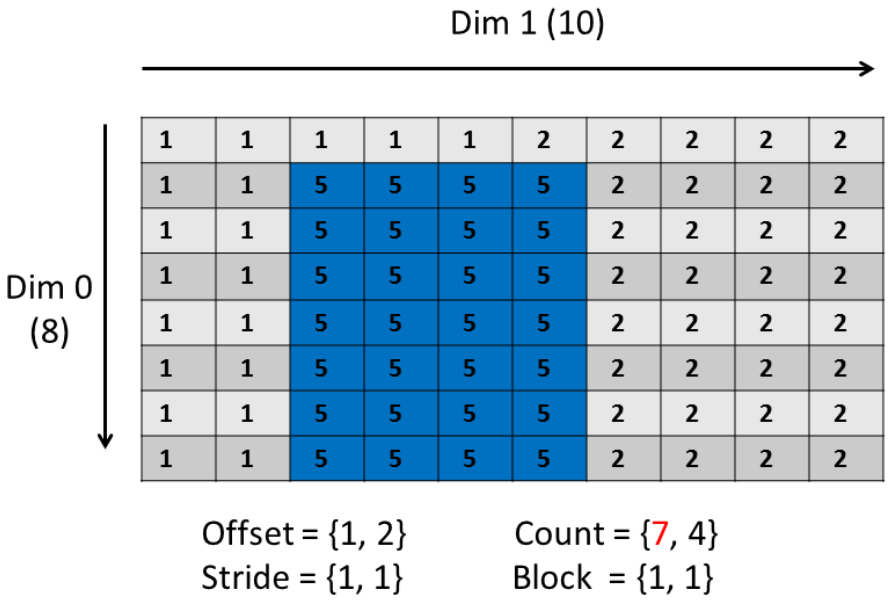
For details on compiling an HDF5 application:

Experiments with Different Selections

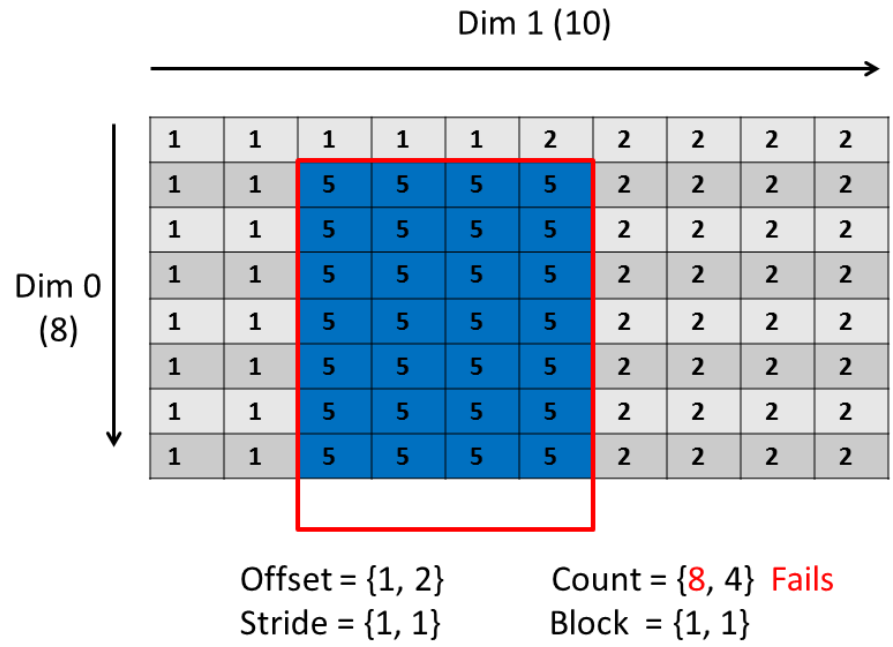
Following are examples of changes that can be made to the example code provided to better understand how to make selections.

Example 1:

By default the example code will select and write to a 3 x 4 subset. You can modify the *count* parameter in the example code to select a different subset, by changing the value of DIM0_SUB (C, C++) / dim0_sub (Fortran) near the top. Change its value to 7 to create a 7 x 4 subset:



If you were to change the subset to 8 x 4, the selection would be beyond the extent of the dimension:



The write will fail with the error: **"file selection+offset not within extent"**

Example 2:

In the example code provided, the memory and file dataspaces passed to the H5Dwrite call have the same size, 3 x 4 (DIM0_SUB x DIM1_SUB). Change the size of the memory dataspace to be 4 x 4 so that they do not match, and then compile:

```
dimsm[0] = DIM0_SUB + 1;
    dimsm[1] = DIM1_SUB;
    memspace_id = H5Screate_simple (RANK, dimsm, NULL);
```

The code will fail with the error: **"src and dest data spaces have different sizes"**

How many elements are in the memory and file dataspaces that were specified above? Add these lines:

```
hssize_t    size;

    /* Just before H5Dwrite call the following */
    size = H5Sget_select_npoints (memspace_id);
    printf ("\nmemspace_id size: %i\n", size);
    size = H5Sget_select_npoints (dataspace_id);
    printf ("dataspace_id size: %i\n", size);
```

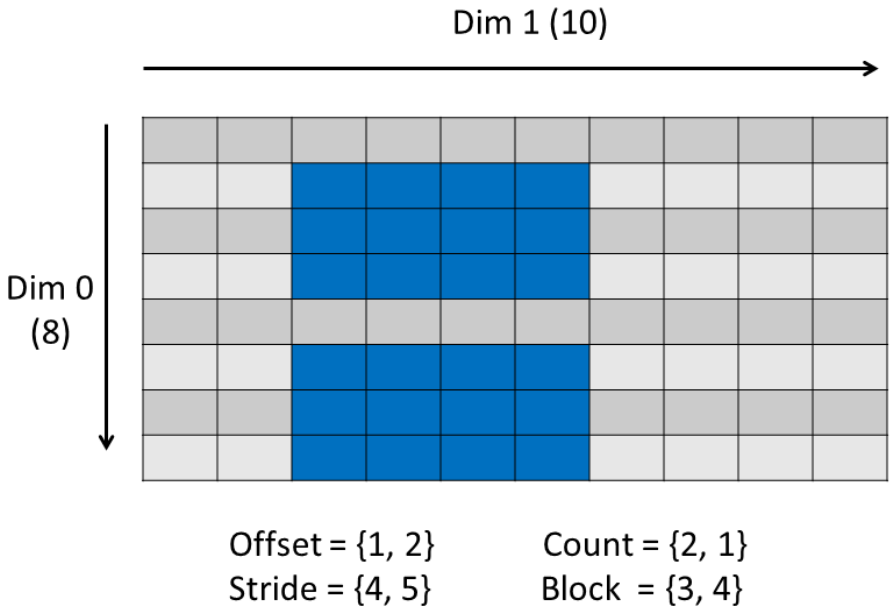
You should see these lines followed by the error:

```
memspace_id size: 16
    dataspace_id size: 12
```

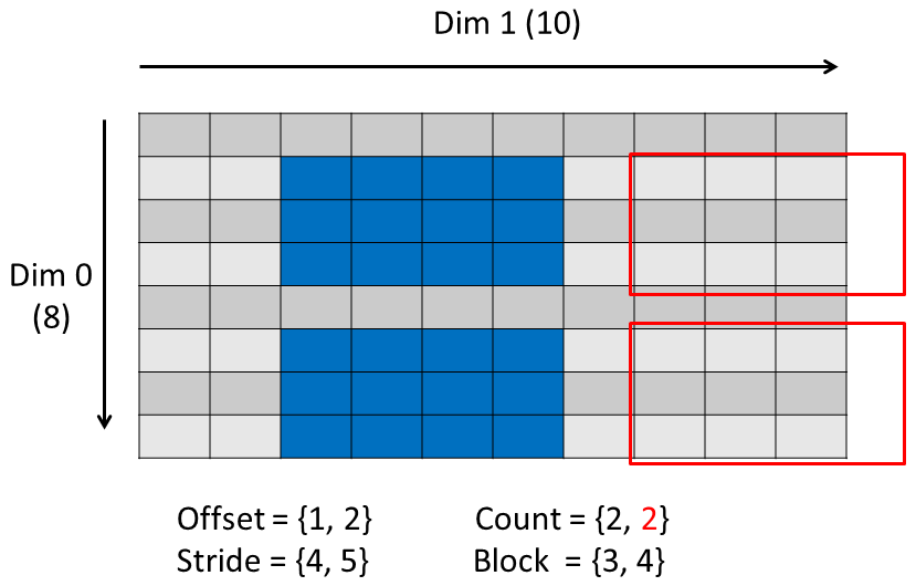
Example 3:

This example shows the selection that occurs if changing the values of the *offset*, *count*, *stride* and *block* parameters in the example code.

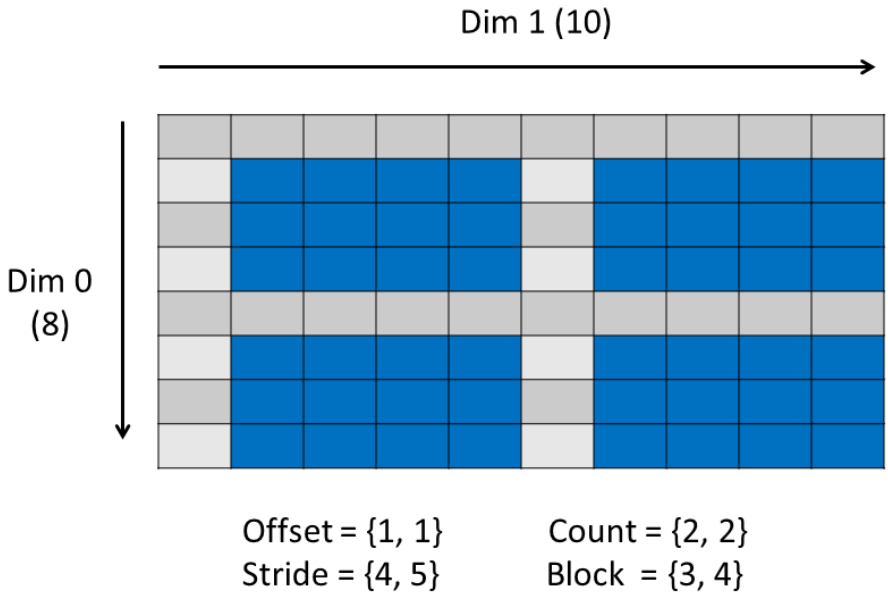
This will select two blocks. The *count* array specifies the number of blocks. The *block* array specifies the size of a block. The *stride* must be modified to accommodate the block size.



Now try modifying the *count* as shown below. The write will fail because the selection goes beyond the extent of the dimension:



If the offset were 1x1 (instead of 1x2), then the selection can be made:



The selections above were tested with the h5_subsetbk.c example code. The memory dataspace was defined as one-dimensional.

Remarks

- In addition to H5S_SELECT_HYPERSLAB, this example introduces the H5D_GET_SPACE call to obtain the dataspace of a dataset.
- If using the default values for the *stride* and *block* parameters of H5S_SELECT_HYPERSLAB, then, for C you can specify NULL for these parameters, rather than passing in an array for each, and for Fortran 90 you can omit these parameters.