UPCIO Testing Strategy v1.2.0-r1

Kun Xi, Ashrujit Mohanty, Tarek El-Ghazawi
The George Washington University
801 22nd Street NW. Suite 607
Washington, DC 20052, USA
{kunxi, ashrujit, tarek}@gwu.edu

Abstract

The goal of this effort is to develop a synthesized UPC-IO library validation suite. It is intended to test the functionality of UPC-IO library implementation and measure the degree of compliance to the UPC-IO specification. This suite contains a set of test cases, which are categorized as the following:

Positive tests: These tests aim to inspect the functionality of UPC-IO library.

Negative tests: These tests are designed to determine the error-detection of UPC-IO.

The section numbers in this testing strategy correspond to the section numbers in the UPC-IO specification v1.2. Each test case verifies its corresponding statement in the UPC-IO specification. Every test program should have a name in the following format: section.subsection[.subsection].casenum.c, while the binary executives are named as section.subsection[.subsection].casenum. All the test cases are run by script/testsuite_run shell-script; this script will collect the exit status and verbose information to generate user-friendly testlog.

Each case has three possible exit statuses:

PASS: This test case exits normally, and the result complies to the corresponding specification.

FAIL: This test case exits normally, but the result violates the corresponding specification.

UNKNOWN: The test case exits abnormally due to unexpected error BEFORE the test case is executed, or the specific test condition is not satisfied. For example, it is uncertain whether there is an outstanding asynchronous operation.

7.3.2 UPC File Operations

7.3.2.1 The upc_all_fopen function

Rule	1
Purpose	To check that upc_all_fopen opens the file specified in fname.
Type	Positive
How	Open a non existent file without UPC_CREATE flag, verify it fails. Then create a new file using upc_all_fopen with UPC_CREATE flag. Close the file, and open the file again without UPC_CREATE flag, verify that the function returns a valid file handle.

Rule	2a
Purpose	To check exactly only one flag of UPC_RDONLY, UPC_WRONLY, or UPC_RDWR and only one flag of UPC_COMMON_FP or UPC_INDIVIDUAL_FP can be used in fopen.
Type	Negative
How	Enumerate all possible combinations of above flags {3C0 | 2C0, 3C1 | 2C0 , 3C2 | 2C0 , 3C3 | 2C0 , 3C0 | 2C1 , 3C1 | 2C1 , 3C2 | 2C1 , 3C3 | 2C1 , 3C0 | 2C2 , 3C1 | 2C2 , 3C2 | 2C2 , 3C3 | 2C2, C is the combinatorial function, | is bitwise OR operation}. Verify that except 3C1 * 2C1, all other combinations would cause fopen fail.

Rule	2b
Purpose	To check all flags have unique bitwise representation.
Type	Positive
How	Perform bitwise AND operation on all possible pair wise combination of the flags and verify that the result is non-zero.

Rule	2c
Purpose	To check that file open access modes work properly.
Type	Negative
How	Open a file with UPC_RDONLY flag, and try to perform a write operation (upc_all_fwrite_local). Verify that the write operation fails. Then open a file with UPC_WRONLY flag and try to perform a read operation (upc_all_fread_local). Verify that the read operation fails. At the end, open a file with UPC_RDWR flag, perform read and write operations. Verify that those operations succeeds.

Rule	2d
Purpose	To check that file open access modes work properly.
Type	Positive
How	Open a file with UPC_RDWR flag, perform read and write operations. Verify that those operations succeeds.

Rule	2e
Purpose	To check that UPC_APPEND sets the initial position of the file pointer to the end of the file
Type	Positive

How	Open a file with UPC_APPEND flag, seek the current position of the file by using (fd, 0, UPC_SEEK_CURR). fseek again from the end of the file (UPC_SEEK_END) for negative offset returned by the previous function. Verify that the current position of the file pointer is at the beginning of the file.

Rule	2g
Purpose	To check that UPC_CREATE flag must be used when using UPC_EXCL flag.
Type	Negative
How	Open a file using UPC_EXCL flag alone without UPC_CREATE flag. Verify that upc_all_fopen fails.

Rule	2i
Purpose	To check that UPC_TRUNC flag truncates the file size to 0
Type	Negative
How	Open a file with UPC_CREATE flag, then write some data to the file, close the file. Reopen the file with UPC_TRUNC flag, and verify the size of file is 0.

Rule	2j
Purpose	To check that UPC_DELETE_ON_CLOSE deletes the file when closed.
Type	Negative
How	Open a file with UPC_CREATE flag, then write some data to the file, close the file. Reopen the file with UPC_DELETE_ON_CLOSE flag and close file. Open the file without UPC_CREATE flag, and verify that upc_all_fopen fails.

Rule	10
Purpose	To check that upc_all_fopen returns single valued error.
Type	Positive
How	Open a non-existent file without UPC_CREATE flag, Verify that upc_all_fopen returns NULL and set the same errno on all threads.

7.3.2.2 The upc_all_fclose function

Rule	1a
Purpose	To check that upc_all_close executes an implicit upc_all_fsync before the close operation.
Type	Positive
How	Open a file with UPC_WRONLY, and write some data to it, then close the file without calling upc_all_fsync. Reopen the file and read the file and verify the date integrity.

Rule	1b
Purpose	To check that upc_all_fclose closes the file specified by file handle.
Type	Negative
How	Open a file and get the file handle, then close the file. Try to perform a read or write operation on the invalid file handle. Verify that those operations fail.

Rule	2
Purpose	To check that upc_all_fclose fails on invalid file handle, and returns single valued error(-1).
Type	Negative
How	Open a file to get a file handle, then close this file. Close this invalid file handle, and verify this function returns -1 on all threads.

7.3.2.4 The upc_all_fseek function

Rule	1
Purpose	To check that upc_all_fseek sets the position of file pointer correctly.
Type	Positive
How	Seek to a particular location in the file (end of file or n bytes from beginning).Write something into the file and verify that data is written at correct place in the file (from the new position of the file pointer).

Rule	2a
Purpose	To check that the offset can be relative to the current position of the file pointer, to the beginning of the file, or to the end of the file.. The offset can be negative, which allows seeking backwards.
Type	Positive
How	Open the file to read, seek to the arbitrary position with UPC_SEEK_SET, UPC_SEEK_CUR and UPC_SEEK_END applied, read one byte from the file to verify the data integrity.

Rule	4
Purpose	To check that in case of common file pointer each thread must specify the same offset and origin.
Type	Positive
How	Open a file with UPC_COMMON_FP flag, Each thread specifies different origin or offset. Verify that upc_all_fseek fails and setup the same errno for all threads.

Rule	5a
Purpose	To check that it is legal to seek past the end of the file.
Type	Positive
How	Seek a certain positive offset amount from the end of the file. Verify that the seek operation is successful.

Rule	5b
Purpose	To check that it is illegal to seek to a negative position in the file.
Type	Negative
How	Seek a negative offset from the beginning of the file. Verify that the function fails and returns -1.

7.3.2.5 The upc_all_fset_size function

Rule	3
Purpose	To check that upc_all_fset_size truncates an existing file when the size parameter specified is less than the size of the existing file.
Type	Positive
How	Open an existing file and determine its size by using upc_all_fseek. Call this function to truncate the file, verify that the updated file size is the same as the size parameter.

Rule	4a
Purpose	To check that the upc_all_fset_size function increases the file size when size parameter is more than the size of the existing file.
Type	Positive
How	Call this function on a file handle with size parameter more than the actual file size. Use upc_all_fseek to verify that the file size is increased as the same as the parameter specified. Read the part of original file and verify the data is not modified.

Rule	7
Purpose	To check that the file pointers remain unchanged when using upc_all_fset_size.
Type	Positive
How	Open a file with UPC_COMMON_FP flag, seek to the end of the file. Use upc_all_fset_size to increase and decrease the file size, Check that the offset of the file pointer does not change. Repeat the above operations for the individual file pointer.

7.3.2.6 The upc_all_fget_size function

Rule	1
Purpose	To check that upc_all_fget_size returns the correct file size.
Type	Positive
How	Open an existing file and seek to the end of the file; call upc_all_fget_size verify that the returned values are the same.

7.3.2.7 The upc_all_fpreallocate function

Rule	3a
Purpose	To check that the upc_all_fpreallocate function increases the file size when size parameter is more than the size of the existing file.
Type	Positive
How	Call this function on a file handle with size parameter more than the actual file size. Use upc_all_fseek to verify that the file size is increased as the same as the parameter specified. Read the part of original file and verify the data is not modified.

Rule	3b
Purpose	To check that upc_all_fpreallocate does nothing when the size parameter specified is less than the size of the existing file.
Type	Positive
How	Open an existing file and determine its size by using upc_all_fseek. Call this function to the file, verify that the file size does not change.

Rule	4
Purpose	To check that upc_all_fpreallocate does not change the individual/common file pointers.
Type	Positive
How	Open a file using common file pointers. Use fseek to seek to a location x bytes from the beginning. Use fpreallocate to allocate new space to the file. Check that the file pointer(s) have the same value as it was before the fpreallocate function. Repeat the same for the individual file pointers

7.3.2.8 The upc_all_fcntl function

Rule	1a
Purpose	To check that upc_all_fcntl can get/set consistence and atomicity
Type	Positive
How	Open a file using the default setting, use upc_all_fcntl with UPC_GET_CA_SEMANTICS and verify that it returns 0; call upc_all_fcntl with UPC_SET_STRONG_CA, UPC_SET_WEAK_CA, and verify that upc_all_fcntl get/set the consistence and atomicity correctly.

Rule	1d
Purpose	To check that upc_all_fcntl can get/set file pointer attribute
Type	Positive
How	Open a file using with UPC_INDIVIDUAL_FP flag applied, use upc_all_fcntl with UPC_GET_FP and verify that it returns UPC_INDIVIDUAL_FP; call upc_all_fcntl with UPC_SET_COMMON_FP and verify the return result of upc_all_fcntl with UPC_GET_FP is UPC_COMMON_FP; repeat the above operations with UPC_SET_INDIVIDUAL_FP and UPC_GET_FP flags to verify the upc_all_fcntl can set/get the type of file pointer.

Rule	1g
Purpose	To check that upc_all_fcntl can get correct flags
Type	Positive
How	Open the file, call upc_all_fcntl with UPC_GET_FL applied, verify the return value is the same as the flag specified in upc_all_fopen. Execute upc_all_fcntl with UPC_SET_STRONG_CA, UPC_SET_WEAK_CA, UPC_SET_COMMON_FP,UPC_SET_INDIVIDUAL_FP, and verify that the return value is correct.

Rule	1h
Purpose	To check that upc_all_fcntl can get the file name.
Type	Positive
How	Call upc_all_fcntl with UPC_GET_FN applied, and verify the correctness of the file name.

Rule	1l
Purpose	To check that this function returns the status of outstanding asynchronous operations. And if the asynchronous operation is completed, this function returns the number of bytes read/written.
Type	Positive
How	Open a file and start an asynchronous operation, check the status of asynchronous operation which is named as test A, then call upc_all_fwait_async to wait for the asynchronous operation complete. Check the status again, which is named test B. If A shows outstanding, B shows no outstanding and return value is the same as upc_all_fwait_async, PASS. If both A and B shows no outstanding and the return value are the same as upc_alll_fwait_asyc, UNKNOWN. Otherwise, FAIL.

Rule	2
Purpose	To check that the function returns single value error in case of error.
Type	Negative
How	Use the function with an invalid file descriptor and verify that the function returns -1

The rest part of the test strategy addresses the read/write operations. The basic read/write compound test plays an important role to bootstraps the whole Read/Write operations test suite. It evaluates functionalities of upc_all_fread_local and upc_all_fwrite_local, then generates a magic file, read.test for further use. In the succeeding test strategy, all read operations applies to read.test, we can check the data integrity since the contents of read.test is deterministic. Then in the write operation test, we can utilize the corresponding read operation for data validation.

For all read/write functions on shared buffer(upc_all_*_shared[_async]), it's quite different for the cases on individual file pointer and common file pointer. The test cases on individual file pointer is named as foo.indiv.num; while the test cases on common file pointer is named as foo.common.num.

Due to the variance of shared buffer's memory layout, we need to test the following memory schemes:

- Round robin(RR)
  the elements are distributed in the round-robin style
- By Block(BB)
  the elements are distributed in the round-robin style but the block size is larger than 1
- Indefinite Block(IB)
  all the elements are stored in one thread.
- Inter-block misalignment
  the elements are distributed in By Block style, the total number of elements is divisible by the block size, but the number of blocks is not divisible by the number of threads.
- Intra-block misalignment
  the elements are distributed in By Block style, the total number of elements is divisible by the block size, but the number of blocks is not divisible by the number of threads.

7.3.3.0 Basic Read/Write compound test

Rule	1
Purpose	To check that upc_all_fread_local and upc_all_fwrite_local work on individual file pointer
Type	Positive
How	Open the file read.test with UPC_CREATE, UPC_RDWR, UPC_TRUNC and UPC_INDIVIDUAL_FP flags, call upc_all_fwrite_local to write number sequence circular from 0 to 127 to the file, and read it back using upc_all_fread_local, verify the data integrity.

7.3.3 Reading Data

7.3.3.1 The upc_all_fread_local function

Rule	1a
Purpose	To check that each thread could read different nmemb*size bytes data at different offset from the file specified to the local buffer, and this function call updates the offset of the file pointer.
Type	Positive
How	Open the file with UPC_INDIVIDUAL_FP, each thread seeks to the different offset, specify different nmemb, size parameter, then read data to the buffer. Verify the data integrity.

Rule	2a
Purpose	To check that this function only works for individual file pointer.
Type	Negative
How	Open a file with UPC_COMMON_FP flag, call this function, and verify it fails.

Rule	2b
Purpose	To check that this function only works for the file handle that is open for reading.
Type	Negative
How	Open a file in the WR_ONLY mode and call this function, and verify it fails.

7.3.3.2 The upc_all_fread_shared function

Rule	common.template
Purpose	This is the test template for this function on common file pointer. To check this function can read arbitrary bytes from the file handle opened with UPC_COMMON_FP flag to the shared buffer. and this function call updates the offset of the file pointer.
Type	Positive
How	Open a file with UPC_COMMON_FP flag, all threads seek to the same offset, then specify the same nmemb, size, block parameter, call this function to read data to the shared buffer and verify the data integrity. Note: blocksize, nmemb, size is defined in the following instances.

Rule	common.1
Purpose	The same as common.template.
Type	Positive
How	The same as common.template. blocksize = 32, nmemb = 1024

Rule	common.2
Purpose	The same as common.template.
Type	Positive
How	The same as common.template. blocksize = 1, nmemb = 1024

Rule	common.3
Purpose	The same as common.template.
Type	Positive

How	The same as common.template. blocksize = 0, nmemb = 1024

Rule	common.4
Purpose	The same as common.template.
Type	Positive
How	The same as common.template. blocksize = 17, nmemb = 1024

Rule	indiv.template
Purpose	This is the test template for this function on individual file pointer. To check this function can read arbitrary bytes from the file handle opened with UPC_INDIVIDUAL_FP flag to the shared buffer, and this function call updates the offset of the file pointer.
Type	Positive
How	Open a file with UPC_INDIVIDUAL_FP flag, each threads seek to the different offset, then specify the different nmemb, size, block parameters, call this function to read data to the shared buffer and verify the data integrity. Note: for thread 0, blocksize, nmemb, size is defined as BLOCKSIZE2, NMEMB2 and sizeof(TYPE2), while other threads take the corresponding parameters from macro BLOCKSIZE, NMEMB, sizeof(TYPE2). All these constants are defined in the following instances.

Rule	indiv.1
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 8, NMEMB = 512, TYPE = int BLOCKSIZE2 = 4, NMEMB2 = 32, TYPE2 = char

Rule	indiv.2
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 8, NMEMB = 512, TYPE = int BLOCKSIZE2 = 0, NMEMB2 = 32, TYPE2 = char

Rule	indiv.3
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 8, NMEMB = 512, TYPE = int BLOCKSIZE2 = 1, NMEMB2 = 32, TYPE2 = char

Rule	indiv.4
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 0, NMEMB = 512, TYPE = int BLOCKSIZE2 = 0, NMEMB2 = 32, TYPE2 = char

Rule	indiv.5
Purpose	The same as indiv.template.

Type	Positive
How	The same as indiv.template. BLOCKSIZE = 0, NMEMB = 512, TYPE = int BLOCKSIZE2 = 1, NMEMB2 = 32, TYPE2 = char

Rule	indiv.6
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 1, NMEMB = 512, TYPE = int BLOCKSIZE2 = 1, NMEMB2 = 32, TYPE2 = char

Rule	indiv.7
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 13, NMEMB = 512, TYPE = int BLOCKSIZE2 = 7, NMEMB2 = 32, TYPE2 = char

Rule	indiv.8
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 11, NMEMB = 512, TYPE = int BLOCKSIZE2 = 0, NMEMB2 = 32, TYPE2 = char

Rule	indiv.9
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 19, NMEMB = 512, TYPE = int BLOCKSIZE2 = 1, NMEMB2 = 32, TYPE2 = char

Rule	2b
Purpose	To check that this function only works for the file handle that is open for reading.
Type	Negative
How	Open a file in the WR_ONLY mode and call this function, and verify it fails.

7.3.4 Writing Data

7.3.4.1 The upc_all_fwrite_local function

Rule	1b
Purpose	To check that each thread can write different nmemb*size bytes data at different offsets from the local buffer to the the file specified, and this function call updates the offset of the file pointer.
Type	Positive
How	Open the file with UPC_INDIVIDUAL_FP flag, each thread seeks to the different offset, specify different nmemb, size parameter, then write data to the file. Read data back to the local buffer to verify the data integrity.

Rule	2
Purpose	To check that this function only works for individual file pointer.
Type	Negative
How	Open a file with UPC_COMMON_FP flag, call this function, and verify it fails.

7.3.4.2 The upc_all_fwrite_shared function

Rule	common.template
Purpose	This is the test template for this function on common file pointer. To check this function can write arbitrary bytes from the local buffer to the file handle opened with UPC_COMMON_FP flag. and this function call updates the offset of the file pointer.
Type	Positive
How	Open a file with UPC_COMMON_FP flag, all threads seek to the same offset, then specify the same nmemb, size, block parameter, call this function to read data to the shared buffer and verify the data integrity. Note: blocksize, nmemb, size is defined in the following instances.

Rule	common.1
Purpose	The same as common.template.
Type	Positive
How	The same as common.template. blocksize = 32, nmemb = 1024

Rule	common.2
Purpose	The same as common.template.
Type	Positive
How	The same as common.template. blocksize = 1, nmemb = 1024

Rule	common.3
Purpose	The same as common.template.
Type	Positive
How	The same as common.template. blocksize = 0, nmemb = 1024

Rule	common.4
Purpose	The same as common.template.
Type	Positive
How	The same as common.template. blocksize = 17, nmemb = 1024

Rule	indiv.template
Purpose	This is the test template for this function on individual file pointer. To check this function can write arbitrary bytes from the local buffer to the file handle opened with UPC_INDIVIDUAL_FP flag. and this function call updates the offset of the file pointer.
Type	Positive

How	Open a file with UPC_INDIVIDUAL_FP flag, each threads seek to the different offset, then specify the different nmemb, size, block parameters, call this function to write data to the file and verify the data integrity. Note: for thread 0, blocksize, nmemb, size is defined as BLOCKSIZE2, NMEMB2 and sizeof(TYPE2), while other threads take the corresponding parameters from macro BLOCKSIZE, NMEMB, sizeof(TYPE2). All these constants are defined in the following instances.

Rule	indiv.1
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 8, NMEMB = 512, TYPE = int BLOCKSIZE2 = 4, NMEMB2 = 32, TYPE2 = char

Rule	indiv.2
Purpose	The same as indiv.template.
Type	Positive
Howd	The same as indiv.template. BLOCKSIZE = 8, NMEMB = 512, TYPE = int BLOCKSIZE2 = 0, NMEMB2 = 32, TYPE2 = char

Rule	indiv.3
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 8, NMEMB = 512, TYPE = int BLOCKSIZE2 = 1, NMEMB2 = 32, TYPE2 = char

Rule	indiv.4
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 0, NMEMB = 512, TYPE = int BLOCKSIZE2 = 0, NMEMB2 = 32, TYPE2 = char

Rule	indiv.5
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 0, NMEMB = 512, TYPE = int BLOCKSIZE2 = 1, NMEMB2 = 32, TYPE2 = char

Rule	indiv.6
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 1, NMEMB = 512, TYPE = int BLOCKSIZE2 = 1, NMEMB2 = 32, TYPE2 = char

Rule	indiv.7
Purpose	The same as indiv.template.
Type	Positive

How	The same as indiv.template. BLOCKSIZE = 13, NMEMB = 512, TYPE = int BLOCKSIZE2 = 7, NMEMB2 = 32, TYPE2 = char

Rule	indiv.8
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 11, NMEMB = 512, TYPE = int BLOCKSIZE2 = 0, NMEMB2 = 32, TYPE2 = char

Rule	indiv.9
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 19, NMEMB = 512, TYPE = int BLOCKSIZE2 = 1, NMEMB2 = 32, TYPE2 = char

7.3.5 List I/O

7.3.5.1 The upc_all_fread_list_local function

Rule	1a
Purpose	To check that each thread reads correct data according to the memvec and fileve from specified file to the local buffer, regardless of whether the current file pointer type is individual or common. And this function call does not update the offset of the file pointer.
Type	Positive
How	Open the file with UPC_INDIVIDUAL_FP flag, initialize memvec and filevec, each thread may pass different parameters to those data structure; call this function to read data and verify data integrity. Check the file pointer offset does not change. Repeat the above operations for common file pointer.

Rule	1b
Purpose	To check that this function only works for the file handle that is open for reading.
Type	Negative
How	Open a file in the WR_ONLY mode and call this function, and verify it fails.

7.3.5.2 The upc_all_fread_list_shared function

Rule	template
Purpose	This is the test template for this function. To check that each thread reads correct data according to the memvec and filevec from specified file to the shared buffer, regardless of whether the current file pointer type is individual or common. And this function call does not update the offset of the file pointer.
Type	Positive
How	Open the file with UPC_INDIVIDUAL_FP flag, call this function to read data and verify data integrity. Check the file pointer offset does not change. Repeat the above operations for common file pointer. Note:memvec and filevec are initialized in the following instances.

Rule	1
Purpose	The same as template
Type	Positive
How	The same as template. blocksize = 32

Rule	2
Purpose	The same as template
Type	Positive
How	The same as template. blocksize = 1

Rule	3
Purpose	The same as template
Type	Positive
How	The same as template. blocksize = 0

7.3.5.3 The upc_all_fwrite_list_local function

Rule	1a
Purpose	To check that each thread writes correct data according to the memvec and fileve from the local buffer to the specified file, regardless of whether the current file pointer type is individual or common. And this function call does not update the offset of the file pointer.
Type	Positive
How	Open the file with UPC_INDIVIDUAL_FP flag, initialize memvec and filevec, each thread may pass different parameters to those data structure; call this function to write data and then read data back to verify data integrity. Check the file pointer offset does not change. Repeat the above operations for common file pointer.

Rule	1b
Purpose	To check that this function only works for the file handle that is open for writing.
Type	Negative
How	Open a file in the RD_ONLY mode and call this function, and verify it fails.

7.3.5.4 The upc_all_fwrite_list_shared function

Rule	template
Purpose	This is the test template for this function. To check that each thread writes correct data according to the memvec and filevec from the shared buffer to the specified file, regardless of whether the current file pointer type is individual or common. And this function call does not update the offset of the file pointer.
Type	Positive

How	Open the file with UPC_INDIVIDUAL_FP flag, call this function to write data and call this function to write data and then read data back to verify data integrity. Check the file pointer offset does not change. Repeat the above operations for common file pointer. Note:memvec and filevec are initialized in the following instances.

Rule	1
Purpose	The same as template
Type	Positive
How	The same as template. blocksize = 32

Rule	2
Purpose	The same as template
Type	Positive
How	The same as template. blocksize = 1

Rule	3
Purpose	The same as template
Type	Positive
How	The same as template. blocksize = 0

Rule	1b
Purpose	To check that this function only works for the file handle that is open for writing.
Type	Negative
How	Open a file in the RD_ONLY mode and call this function, and verify it fails.

7.3.6 Asynchronous I/O

Asynchronous I/O operations share the same prototype as synchronous I/O operations, the asynchronous read/write test cases are the same as the corresponding synchronous functions. These test cases are list here just for your convenience.

7.3.6.1 The upc_all_fread_local_async function

Rule	1a
Purpose	To check that each thread could read different nmemb*size bytes data at different offset from the file specified to the local buffer, and this function call updates the offset of the file pointer.
Type	Positive
How	Open the file with UPC_INDIVIDUAL_FP, each thread seeks to the different offset, specify different nmemb, size parameter, then read data to the buffer. Verify the data integrity.

Rule	2a
Purpose	To check that this function only works for individual file pointer.
Type	Negative
How	Open a file with UPC_COMMON_FP flag, call this function, and verify it fails.

Rule	2b
Purpose	To check that this function only works for the file handle that is open for reading.
Type	Negative
How	Open a file in the WR_ONLY mode and call this function, and verify it fails.

7.3.6.2 The upc_all_fread_shared_async function

Rule	common.template
Purpose	This is the test template for this function on common file pointer. To check this function can read arbitrary bytes from the file handle opened with UPC_COMMON_FP flag to the shared buffer. and this function call updates the offset of the file pointer.
Type	Positive
How	Open a file with UPC_COMMON_FP flag, all threads seek to the same offset, then specify the same nmemb, size, block parameter, call this function to read data to the shared buffer and verify the data integrity. Note: blocksize, nmemb, size is defined in the following instances.

Rule	common.1
Purpose	The same as common.template.
Type	Positive
How	The same as common.template. blocksize = 32, nmemb = 1024

Rule	common.2
Purpose	The same as common.template.
Type	Positive
How	The same as common.template. blocksize = 1, nmemb = 1024

Rule	common.3
Purpose	The same as common.template.
Type	Positive
How	The same as common.template. blocksize = 0, nmemb = 1024

Rule	common.4
Purpose	The same as common.template.
Type	Positive
How	The same as common.template. blocksize = 17, nmemb = 1024

Rule	indiv.template
Purpose	This is the test template for this function on individual file pointer. To check this function can read arbitrary bytes from the file handle opened with UPC_INDIVIDUAL_FP flag to the shared buffer, and this function call updates the offset of the file pointer.
Type	Positive

How	Open a file with UPC_INDIVIDUAL_FP flag, each threads seek to the different offset, then specify the different nmemb, size, block parameters, call this function to read data to the shared buffer and verify the data integrity. Note: for thread 0, blocksize, nmemb, size is defined as BLOCKSIZE2, NMEMB2 and sizeof(TYPE2), while other threads take the corresponding parameters from macro BLOCKSIZE, NMEMB, sizeof(TYPE2). All these constants are defined in the following instances.

Rule	indiv.1
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 8, NMEMB = 512, TYPE = int BLOCKSIZE2 = 4, NMEMB = 32, TYPE = char

Rule	indiv.2
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 8, NMEMB = 512, TYPE = int BLOCKSIZE2 = 0, NMEMB = 32, TYPE = char

Rule	indiv.3
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 8, NMEMB = 512, TYPE = int BLOCKSIZE2 = 1, NMEMB = 32, TYPE = char

Rule	indiv.4
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 0, NMEMB = 512, TYPE = int BLOCKSIZE2 = 0, NMEMB = 32, TYPE = char

Rule	indiv.5
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 0, NMEMB = 512, TYPE = int BLOCKSIZE2 = 1, NMEMB = 32, TYPE = char

Rule	indiv.6
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 1, NMEMB = 512, TYPE = int BLOCKSIZE2 = 1, NMEMB = 32, TYPE = char

Rule	indiv.7
Purpose	The same as indiv.template.
Type	Positive

How	The same as indiv.template. BLOCKSIZE = 13, NMEMB = 512, TYPE = int BLOCKSIZE2 = 7, NMEMB = 32, TYPE = char

Rule	indiv.8
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 11, NMEMB = 512, TYPE = int BLOCKSIZE2 = 0, NMEMB = 32, TYPE = char

Rule	indiv.9
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 19, NMEMB = 512, TYPE = int BLOCKSIZE2 = 1, NMEMB = 32, TYPE = char

Rule	2b
Purpose	To check that this function only works for the file handle that is open for reading.
Type	Negative
How	Open a file in the WR_ONLY mode and call this function, and verify it fails.

7.3.6.3 The upc_all_fwrite_local_async function

Rule	1b
Purpose	To check that each thread can write different nmemb*size bytes data at different offsets from the local buffer to the the file specified, and this function call updates the offset of the file pointer.
Type	Positive
How	Open the file with UPC_INDIVIDUAL_FP flag, each thread seeks to the different offset, specify different nmemb, size parameter, then write data to the file. Read data back to the local buffer to verify the data integrity.

Rule	2
Purpose	To check that this function only works for individual file pointer.
Type	Negative
How	Open a file with UPC_COMMON_FP flag, call this function, and verify it fails.

7.3.6.4 The upc_all_fwrite_shared_async function

Rule	common.template
Purpose	This is the test template for this function on common file pointer. To check this function can write arbitrary bytes from the local buffer to the file handle opened with UPC_COMMON_FP flag. and this function call updates the offset of the file pointer.
Type	Positive
How	Open a file with UPC_COMMON_FP flag, all threads seek to the same offset, then specify the same nmemb, size, block parameter, call this function to read data to the shared buffer and verify the data integrity. Note: blocksize, nmemb, size is defined in the following instances.

Rule	common.1
Purpose	The same as common.template.
Type	Positive
How	The same as common.template. blocksize = 32, nmemb = 1024

Rule	common.2
Purpose	The same as common.template.
Type	Positive
How	The same as common.template. blolscksize = 1, nmemb = 1024

Rule	common.3
Purpose	The same as common.template.
Type	Positive
How	The same as common.template. blocksize = 0, nmemb = 1024

Rule	common.4
Purpose	The same as common.template.
Type	Positive
How	The same as common.template. blocksize = 17, nmemb = 1024

Rule	indiv.template
Purpose	This is the test template for this function on individual file pointer. To check this function can write arbitrary bytes from the local buffer to the file handle opened with UPC_INDIVIDUAL_FP flag. and this function call updates the offset of the file pointer.
Type	Positive
How	Open a file with UPC_INDIVIDUAL_FP flag, each threads seek to the different offset, then specify the different nmemb, size, block parameters, call this function to write data to the file and verify the data integrity. Note: for thread 0, blocksize, nmemb, size is defined as BLOCKSIZE2, NMEMB2 and sizeof(TYPE2), while other threads take the corresponding parameters from macro BLOCKSIZE, NMEMB, sizeof(TYPE2). All these constants are defined in the following instances.

Rule	indiv.1
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 8, NMEMB = 512, TYPE = int BLOCKSIZE2 = 4, NMEMB = 32, TYPE = char

Rule	indiv.2
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 8, NMEMB = 512, TYPE = int BLOCKSIZE2 = 0, NMEMB = 32, TYPE = char

Rule	indiv.3
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 8, NMEMB = 512, TYPE = int BLOCKSIZE2 = 1, NMEMB = 32, TYPE = char

Rule	indiv.4
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 0, NMEMB = 512, TYPE = int BLOCKSIZE2 = 0, NMEMB = 32, TYPE = char

Rule	indiv.5
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 0, NMEMB = 512, TYPE = int BLOCKSIZE2 = 1, NMEMB = 32, TYPE = char

Rule	indiv.6
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 1, NMEMB = 512, TYPE = int BLOCKSIZE2 = 1, NMEMB = 32, TYPE = char

Rule	indiv.7
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 13, NMEMB = 512, TYPE = int BLOCKSIZE2 = 7, NMEMB = 32, TYPE = char

Rule	indiv.8
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 11, NMEMB = 512, TYPE = int BLOCKSIZE2 = 0, NMEMB = 32, TYPE = char

Rule	indiv.9
Purpose	The same as indiv.template.
Type	Positive
How	The same as indiv.template. BLOCKSIZE = 19, NMEMB = 512, TYPE = int BLOCKSIZE2 = 1, NMEMB = 32, TYPE = char

7.3.6.5 The upc_all_fread_list_local_async function

Rule	1a
Purpose	To check that each thread reads correct data according to the memvec and fileve from specified file to the local buffer, regardless of whether the current file pointer type is individual or common. And this function call does not update the offset of the file pointer.
Type	Positive
How	Open the file with UPC_INDIVIDUAL_FP flag, initialize memvec and filevec, each thread may pass different parameters to those data structure; call this function to read data and verify data integrity. Check the file pointer offset does not change. Repeat the above operations for common file pointer.

Rule	1b
Purpose	To check that this function only works for the file handle that is open for reading.
Type	Negative
How	Open a file in the WR_ONLY mode and call this function, and verify it fails.

7.3.6.6 The upc_all_fread_list_shared_async function

Rule	template
Purpose	This is the test template for this function. To check that each thread reads correct data according to the memvec and filevec from specified file to the shared buffer, regardless of whether the current file pointer type is individual or common. And this function call does not update the offset of the file pointer.
Type	Positive
How	Open the file with UPC_INDIVIDUAL_FP flag, call this function to read data and verify data integrity. Check the file pointer offset does not change. Repeat the above operations for common file pointer. Note:memvec and filevec are initialized in the following instances.

Rule	1
Purpose	The same as template
Type	Positive
How	The same as template. blocksize = 32

Rule	2
Purpose	The same as template
Type	Positive
How	The same as template. blocksize = 1

Rule	3
Purpose	The same as template
Type	Positive
How	The same as template. blocksize = 0

7.3.6.7 The upc_all_fwrite_list_local_async function

Rule	1a
Purpose	To check that each thread writes correct data according to the memvec and fileve from the local buffer to the specified file, regardless of whether the current file pointer type is individual or common. And this function call does not update the offset of the file pointer.
Type	Positive
How	Open the file with UPC_INDIVIDUAL_FP flag, initialize memvec and filevec, each thread may pass different parameters to those data structure; call this function to write data and then read data back to verify data integrity. Check the file pointer offset does not change. Repeat the above operations for common file pointer.

Rule	1b
Purpose	To check that this function only works for the file handle that is open for writing.
Type	Negative
How	Open a file in the RD_ONLY mode and call this function, and verify it fails.

7.3.6.8 The upc_all_fwrite_list_shared_async function

Rule	template
Purpose	This is the test template for this function. To check that each thread writes correct data according to the memvec and filevec from the shared buffer to the specified file, regardless of whether the current file pointer type is individual or common. And this function call does not update the offset of the file pointer.
Type	Positive
How	Open the file with UPC_INDIVIDUAL_FP flag, call this function to write data and call this function to write data and then read data back to verify data integrity. Check the file pointer offset does not change. Repeat the above operations for common file pointer. Note:memvec and filevec are initialized in the following instances.

Rule	1
Purpose	The same as template
Type	Positive
How	The same as template. blocksize = 32

Rule	2
Purpose	The same as template
Type	Positive
How	The same as template. blocksize = 1

Rule	3
Purpose	The same as template
Type	Positive

How	The same as template. blocksize = 0

Rule	1b
Purpose	To check that this function only works for the file handle that is open for writing.
Type	Negative
How	Open a file in the RD_ONLY mode and call this function, and verify it fails.

7.3.6.9 The upc_all_fwait_async function

upc_all_fwait_async is called to complete the outstanding asynchronous operation. It has been tested in the above test cases. Please refer to *7.3.6.10.2* for upc_all_ftest_async/upc_all_fwait compound test, and *7.2.2.8.1l* for upc_all_fntl/upc_all_fwait_async

7.3.6.10 The upc_all_ftest_async function

Rule	1
Purpose	To check that upc_all_ftest_async will, eventually set the flag to 1, i.e the outstanding asynchronous operation has completed.
Type	Positive
How	Open a file and start an asynchronous operation, poll the status of asynchronous operation periodically. If in the designated time period, the flag is set to 1, this function passes, otherwise it fails.

Rule	2
Purpose	To check that this function returns the status of outstanding asynchronous operations. And if the asynchronous operation is completed, this function returns the number of bytes read/written.
Type	Positive
How	Open a file and start an asynchronous operation, check the status of asynchronous operation which is named as test A, then call upc_all_fwait_async to wait for the asynchronous operation complete. Check the status again, which is named test B. If A shows outstanding, B shows no outstanding and return value is the same as upc_all_fwait_async, PASS. If both A and B shows no outstanding and the return value are the same as upc_alll_fwait_asyc, UNKNOWN. Otherwise, FAIL.