CLAIMS

1. An inter-process data transmission method, comprising:

after a local inter-process communication connection is created for a first program process and a second program process, acquiring a data stream identifier through the first program process;

writing, through the first program process, target data into a data buffer of a shared memory, and writing a first read/write status value into a target location of a first bitmap field of the shared memory based on the data stream identifier after the target data is written, wherein the target location is determined based on the data stream identifier; and

performing, through the second program process, busy-wait polling detection on the first bitmap field of the shared memory, and if the first read/write status value is detected at the target location of the first bitmap field, acquiring, from the data buffer, the target data output by the first program process;

wherein the shared memory comprises a data length field into which a data length of the target data is written, and wherein the data length field and the data buffer are arranged continuously in the shared memory after the first bitmap field, and an interval in the shared memory exists between the first bitmap field and the data length field.

2. The method according to claim 1, wherein the acquiring the data stream identifier through the first program process comprises:

sending, via the first program process, a handshake request message to the second program process through the local inter-process communication connection, wherein the handshake request message is used for registering a data stream transmission task on a side of the second program process;

processing, through the second program process, the handshake request message to generate a handshake response message, and returning the handshake response message to the first program process, wherein the handshake response message comprises the data stream identifier corresponding to the data stream transmission task; and

receiving, through the first program process, the handshake response message to obtain the data stream identifier.

3. The method according to claim 2, wherein the handshake request message comprises a first request field and/or a second request field, the first request field is used for determining program version information of the second program process adapted to the first program process, and the second request field is used for representing

a memory space required by the data stream transmission task.

4. The method according to claim 2, wherein the handshake response message further comprises at least one of the following:

a first response field, which is used for indicating a maximum concurrency of the second program process for executing the data stream transmission task;

a second response field, which is used for indicating a memory capacity of the shared memory; and

a third response field, which is used for indicating a bitmap field in the shared memory corresponding to the data stream transmission task.

5. The method according to claim 1, wherein the writing, through the first program process, the target data into the data buffer of the shared memory comprises:

acquiring an offset address of the data buffer; and

determining the data buffer according to the offset address, and writing the target data into the data buffer.

6. The method according to claim 5, further comprising:

acquiring the data length of the target data; and

writing the data length into the data length field before the data buffer; and

the acquiring, from the data buffer, the target data output by the first program process comprises:

acquiring, from the data length field of the shared memory, a target data length corresponding to the data stream identifier; and

reading the data buffer based on the target data length to obtain the target data.

7. The method according to claim 1, further comprising:

after the first program process meets a stop condition, closing the local inter-process communication connection on a side of the first program process; and

when the second program process listens that the first program process closes the local inter-process communication connection, closing the local inter-process communication connection on a side of the second program process.

8. The method according to claim 7, wherein the closing the local inter-process communication connection on the side of the second program process comprises:

sequentially performing a first release step and a second release step to close the local inter-process communication connection on the side of the second program process,

wherein the first release step is used for writing a notify event file descriptor and closing the notify event file descriptor subsequently, and the notify event file descriptor is used for notifying the first program process

that data transmission by the second program process is completed; and

the second release step is used for setting data at target locations of all bitmap fields to a second status value, and cleaning related resources of a data stream.

9. The method according to claim 7, before the closing the local inter-process communication connection on the side of the first program process, further comprising:

writing, through the first program process, a first application close status value into a target location of a second bitmap field of the shared memory, wherein the first application close status value is used for triggering closing of the local inter-process communication connection for the second program process.

10. The method according to claim 1, further comprising:

after the second program process meets a stop condition, acquiring a connection state of the local inter-process communication connection for the first program process;

if the local inter-process communication connection for the first program process is closed, directly closing the local inter-process communication connection on a side of the second program process; or

if the local inter-process communication connection for the first program process is not closed, triggering closing of the local inter-process communication connection for the first program process, and setting the second program process to a wait-to-close state; and closing the local inter-process communication connection on a side of the second program process after the local inter-process communication connection for the first program process is closed.

11. The method according to claim 10, wherein the triggering the closing of the local inter-process communication connection for the first program process comprises:

writing, through the second program process, a first service close status value into a target location of a second bitmap field of the shared memory; and

when the first program process detects that the target location of the second bitmap field of the shared memory is the first service close status value, triggering a stop condition for closing the local inter-process communication connection, and writing a first application close status value into a target location of a third bitmap field of the shared memory; and

the closing the local inter-process communication connection on the side of the second program process after the local inter-process communication connection for the first program process is closed comprises:

when the second program process detects that the target location of the third bitmap field of the shared memory is the first application close status value, or after a preset duration elapses, closing the local inter-process communication connection on the side of the second program process.

12. An inter-process data transmission apparatus, comprising:

an acquiring module, configured to: after a local inter-process communication connection is created for a first program process and a second program process, acquire a data stream identifier through the first program process;

a writing module, configured to: write, through the first program process, target data into a data buffer of a shared memory, and write a first read/write status value into a target location of a first bitmap field of the shared memory based on the data stream identifier after the target data is written, wherein the target location is determined based on the data stream identifier; and

a reading module, configured to: perform busy-wait polling detection on the first bitmap field of the shared memory through the second program process, and if the first read/write status value is detected at the target location of the first bitmap field, acquire, from the data buffer, the target data output by the first program process;

wherein the shared memory comprises a data length field into which a data length of the target data is written, and wherein the data length field and the data buffer are arranged continuously in the shared memory after the first bitmap field, and an interval in the shared memory exists between the first bitmap field and the data length field.

13. The inter-process data transmission apparatus according to claim 12, wherein the acquiring module is configured to:

send, via the first program process, a handshake request message to the second program process through the local inter-process communication connection, wherein the handshake request message is used for registering a data stream transmission task on a side of the second program process;

process, through the second program process, the handshake request message to generate a handshake response message, and return the handshake response message to the first program process, wherein the handshake response message comprises the data stream identifier corresponding to the data stream transmission task; and

receive, through the first program process, the handshake response message to obtain the data stream identifier.

14. The inter-process data transmission apparatus according to claim 13, wherein the handshake request message comprises a first request field and/or a second request field, the first request field is used for determining program version information of the second program process adapted to the first program process, and the second request field is used for representing a memory space required by the data stream transmission task.

15. The inter-process data transmission apparatus according to claim 13, wherein the handshake response

message further comprises at least one of the following:

a first response field, which is used for indicating a maximum concurrency of the second program process for executing the data stream transmission task;

a second response field, which is used for indicating a memory capacity of the shared memory; and

a third response field, which is used for indicating a bitmap field in the shared memory corresponding to the data stream transmission task.

16. The inter-process data transmission apparatus according to claim 12, wherein the writing module is configured to:

acquire an offset address of the data buffer; and

determine the data buffer according to the offset address, and write the target data into the data buffer.

17. The inter-process data transmission apparatus according to claim 16, wherein the writing module is further configured to:

acquire the data length of the target data; and

write the data length into the data length field before the data buffer; and

the reading module is configured to:

acquire, from the data length field of the shared memory, a target data length corresponding to the data stream identifier; and

read the data buffer based on the target data length to obtain the target data.

18. An electronic device, comprising: a processor and a memory,

wherein the memory stores a computer-executable instruction; and

the processor executes the computer-executable instruction stored in the memory, to enable the processor to execute the inter-process data transmission method according to any one of claims 1 to 11.

19. A computer-readable storage medium storing a computer-executable instruction therein, wherein when a processor executes the computer-executable instruction, the inter-process data transmission method according to any one of claims 1 to 11 is implemented.

20. A computer program product, comprising a computer program, wherein when the computer program is executed by a processor, the inter-process data transmission method according to any one of claims 1 to 11 is implemented.