NAME

curl_multi_perform - reads/writes available data from each easy handle

SYNOPSIS

```
#include <curl/curl.h>

CURLMcode curl_multi_perform(CURLM *multi_handle, int *running_handles);
```

DESCRIPTION

When the app thinks there's data available for the multi_handle, it should call this function to read/write whatever there is to read or write right now. curl_multi_perform() returns as soon as the reads/writes are done. This function does not require that there actually is any data available for reading or that data can be written, it can be called just in case. It will write the number of handles that still transfer data in the second argument's integer-pointer.

When you call curl_multi_perform() and the amount of *running_handles* is changed from the previous call (or is less than the amount of easy handles you've added to the multi handle), you know that there is one or more transfers less "running". You can then call *curl_multi_info_read(3)* to get information about each individual completed transfer, and that returned info includes CURLcode and more.

RETURN VALUE

CURLMcode type, general libcurl multi interface error code.

If you receive *CURLM_CALL_MULTI_PERFORM*, this basically means that you should call *curl_multi_perform* again, before you select() on more actions. You don't have to do it immediately, but the return code means that libcurl may have more data available to return or that there may be more data to send off before it is "satisfied". Do note that *curl_multi_perform(3)* will return *CURLM_CALL_MULTI_PERFORM* only when it wants to be called again **immediately**. When things are fine and there are nothing immediate it wants done, it'll return *CURLM_OK* and you need to wait for "action" and then call this function again.

NOTE that this only returns errors etc regarding the whole multi stack. There might still have occurred problems on individual transfers even when this function returns *CURLM_OK*.

TYPICAL USAGE

Most applications will use *curl_multi_fdset(3)* to get the multi_handle's file descriptors, then it'll wait for action on them using **select(3)** and as soon as one or more of them are ready, *curl_multi_perform(3)* gets called.

SEE ALSO

curl_multi_cleanup(3), **curl_multi_init**(3), **curl_multi_fdset**(3), **curl_multi_info_read**(3), **libcurl-errors**(3)