NAME

curl_multi_waitfds − extract file descriptor information from a multi handle

SYNOPSIS

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

CURLMcode curl_multi_waitfds(CURLM *multi,
                             struct curl_waitfd *ufds,
                             unsigned int size,
                             unsigned int *fd_count);
```

DESCRIPTION

This function extracts *curl_waitfd* structures which are similar to *poll(2)*'s *pollfd* structure from a given multi_handle.

These structures can be used for polling on multi_handle file descriptors in a fashion similar to *curl_multi_poll(3)*. The *curl_multi_perform(3)* function should be called as soon as one of them is ready to be read from or written to.

libcurl fills provided *ufds* array up to the *size*. If a number of descriptors used by the multi_handle is greater than the *size* parameter then libcurl returns CURLM_OUT_OF_MEMORY error.

If the *fd_count* argument is not a null pointer, it points to a variable that on return specifies the number of descriptors used by the multi_handle to be checked for being ready to read or write.

The client code can pass *size* equal to zero to get the number of the descriptors and allocate appropriate storage for them to be used in a subsequent function call. In this case, *fd_count* receives a number greater than or equal to the number of descriptors.

PROTOCOLS

This functionality affects all supported protocols

EXAMPLE

```
#include <stdlib.h>

int main(void)
{
 CURLMcode mresult;
 struct curl_waitfd *ufds;

 CURLM *multi = curl_multi_init();

 do {
  /* call curl_multi_perform() */

  /* get the count of file descriptors from the transfers */
  unsigned int fd_count = 0;

  mresult = curl_multi_waitfds(multi, NULL, 0, &fd_count);

  if(mresult != CURLM_OK) {
   fprintf(stderr, "curl_multi_waitfds() failed, code %d.\n", mresult);
   break;
  }

  if(!fd_count)
```

```
    continue; /* no descriptors yet */

  /* allocate storage for our descriptors */
  ufds = malloc(fd_count * sizeof(struct curl_waitfd));

  /* get wait descriptors from the transfers and put them into array. */
  mresult = curl_multi_waitfds(multi, ufds, fd_count, &fd_count);

  if(mresult != CURLM_OK) {
    fprintf(stderr, "curl_multi_waitfds() failed, code %d.\n", mresult);
    free(ufds);
    break;
  }

  /* Do polling on descriptors in ufds */

  free(ufds);
 } while(!mresult);
}
```

AVAILABILITY

Added in curl 8.8.0

RETURN VALUE

This function returns a CURLMcode indicating success or error.

CURLM_OK (0) means everything was OK, non−zero means an error occurred, see *libcurl−errors(3)*.

SEE ALSO

curl_multi_fdset(3), **curl_multi_perform**(3), **curl_multi_poll**(3), **curl_multi_wait**(3)