

**NAME**

ares\_process – Process events for name resolution

**SYNOPSIS**

```
#include <ares.h>
```

```
void ares_process(ares_channel channel, fd_set *read_fds,  
fd_set *write_fds)
```

```
void ares_process_fd(ares_channel channel, ares_socket_t read_fd, ares_socket_t write_fd)
```

**DESCRIPTION**

The **ares\_process(3)** function handles input/output events and timeouts associated with queries pending on the name service channel identified by *channel*. The file descriptor sets pointed to by *read\_fds* and *write\_fds* should have file descriptors set in them according to whether the file descriptors specified by *ares\_fds(3)* are ready for reading and writing. (The easiest way to determine this information is to invoke **select** with a timeout no greater than the timeout given by *ares\_timeout(3)* ).

The **ares\_process** function will invoke callbacks for pending queries if they complete successfully or fail.

**ares\_process\_fd(3)** works the same way but acts and operates only on the specific file descriptors (sockets) you pass in to the function. Use **ARES\_SOCKET\_BAD** for "no action". This function is of course provided to allow users of c-ares to void **select()** in their applications and within c-ares.

**EXAMPLE**

The following code fragment waits for all pending queries on a channel to complete:

```
int nfds, count;  
fd_set readers, writers;  
struct timeval tv, *tvp;  
  
while (1)  
{  
    FD_ZERO(&readers);  
    FD_ZERO(&writers);  
    nfds = ares_fds(channel, &readers, &writers);  
    if (nfds == 0)  
        break;  
    tvp = ares_timeout(channel, NULL, &tv);  
    count = select(nfds, &readers, &writers, NULL, tvp);  
    ares_process(channel, &readers, &writers);  
}
```

**SEE ALSO**

**ares\_fds(3)**, **ares\_timeout(3)**

**AUTHOR**

Greg Hudson, MIT Information Systems  
Copyright 1998 by the Massachusetts Institute of Technology.