NAME

w_rc_t − Shore return code

SYNOPSIS

```
#include <w.h>
#include <w_rc.h>
class w_rc_t;
class w_rc_i;
```

DESCRIPTION

The class *w_rc_t* encapsulates integer error codes, prints error messages for the error codes, lets you build and print stack traces when error codes are returned, and prints an error message if the error codes are not evaluated (checked for errors).

If a function returns a *w_rc_t* that is ignored, the *w_rc_t* prints the message
`Error not checked`
when the instance is destroyed. The error message is printed by the **error_not_checked** method, making it a useful debugging breakpoint. The *w_rc_t* maintain reference counts on the various *w_error_t* structures for the purpose of collecting garbage.

MACROS

Use the following macros to create and manipulate return codes:

rc = RC(e)
: Constructs an instance of *w_rc_t* containing the current line and file information and the error code *e.* If *e* is the constant *w_error_t::no_error,* the resulting return code's Boolean conversion operator returns *false,* otherwise it returns *true.* returns

RCOK
: Constructs an instance of *w_rc_t* that signifies no error; equivalent to
```
w_rc_t(w_error_t::no_error);
```

rc = RC_AUGMENT(rc)
: Adds the current line number and file name to the information stored in *rc.*

rc = RC_PUSH(rc, e)
: Pushes a new error code, *e,* onto the stack along with the current line number and file name.

DISPLAYING ERROR MESSAGES

Printing an instance of *w_rc_t* descriptive string for the error code to be printed, along with the stack trace (line numbers and file names).

```
...
return RC(eUSERABORT);

...
w_rc_t rc = RC(eUSERABORT);

cerr << rc << endl;
```

REFERENCE COUNTING

References to *w_rc_t* structures are counted. Return codes that are destroyed without being checked cause this message to be printed to the standard error stream:
`Error not checked`

Checking a return code amounts to seeing if its Boolean conversion operator is true or false:

```
// call method z, which returns
// an instance of w_rc_t
rc = z();
if(rc) {
       // error case
} else {
```

```
        // z() returned RCOK
}
```

FORGETTING TO RETURN A W _RC_T

Sometime a function that returns a **w_rc_t** will not due so due to a bug. The compiler should catch this, but we've seen gcc miss it. As a result, your program may crash with a stack trace something like this (from gdb):

```
#0  0xef7991dc in strrchr ()
#1  0x44a0 in __ls__FR7ostreamRC9w_error_t (o=@0x10c33c, obj=@0xa000)
    at w_error.c:197
#2  0xa8728 in __ls__FR7ostreamRC6w_rc_t (o=@0x10c33c, obj=@0xeffff840)
    at w_rc.c:81
#3  0xa8690 in fatal__6w_rc_t (this=0xeffff840) at w_rc.c:56
#4  0x3ba4 in main (argc=-268437392, argv=0xefffff914) at hello.C:154
```

RC-LITE

When Shore is configured with -DCHEAP _RC (not the default), a lighter-weight implementation of the class *w_rc_t* is used (it is found in `#include <w_cheaprc.h>`) in order to reduce processing costs. In this case, all the macros described here are defined, but many of them do nothing, since the cheaper implementation of the class maintains no stack trace, line numbers, or file names. All it stores is a single integer representing a single error, and it does no reference-counting.

VERSION

This manual page applies to Version 2.0 of the Shore Storage Manager.

SPONSORSHIP

The Shore project is sponsored by the Advanced Research Project Agency, ARPA order number 018 (formerly 8230), monitored by the U.S. Army Research Laboratory under contract DAAB07-91-C-Q518. Further funding for this work was provided by DARPA through Rome Research Laboratory Contract No. F30602-97-2-0247.

COPYRIGHT

SEE ALSO

error(fc) and **intro(fc).**