NAME

reallocarray - memory reallocation function

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <stdlib.h>

*void **

reallocarray(*void *ptr, size_t nmemb, size_t size*);

DESCRIPTION

The **reallocarray**() function is similar to the **realloc**() function except it operates on *nmemb* members of size *size* and checks for integer overflow in the calculation *nmemb* * *size*.

RETURN VALUES

The **reallocarray**() function returns a pointer to the allocated space; otherwise, a NULL pointer is returned and *errno* is set to ENOMEM.

EXAMPLES

Consider **reallocarray**() when there is multiplication in the *size* argument of **malloc**() or **realloc**(). For example, avoid this common idiom as it may lead to integer overflow:

```
if ((p = malloc(num * size)) == NULL)
        err(1, "malloc");
```

A drop-in replacement is the OpenBSD extension **reallocarray**():

```
if ((p = reallocarray(NULL, num, size)) == NULL)
        err(1, "reallocarray");
```

When using **realloc**(), be careful to avoid the following idiom:

```
size += 50;
if ((p = realloc(p, size)) == NULL)
        return (NULL);
```

Do not adjust the variable describing how much memory has been allocated until the allocation has been successful. This can cause aberrant program behavior if the incorrect size value is used. In most cases,

the above sample will also result in a leak of memory. As stated earlier, a return value of NULL indicates that the old object still remains allocated. Better code looks like this:

```
newsize = size + 50;
if ((newp = realloc(p, newsize)) == NULL) {
        free(p);
        p = NULL;
        size = 0;
        return (NULL);
}
p = newp;
size = newsize;
```

As with **malloc**(), it is important to ensure the new size value will not overflow; i.e. avoid allocations like the following:

```
if ((newp = realloc(p, num * size)) == NULL) {
        ...
```

Instead, use **reallocarray**():

```
if ((newp = reallocarray(p, num, size)) == NULL) {
        ...
```

SEE ALSO

realloc(3)

HISTORY

The **reallocarray**() function first appeared in OpenBSD 5.6 and FreeBSD 11.0.