

Bit-masking is a very useful method to emulate Unix-style file permissions (read/write/execute for example). What's nice about a PHP implementation is that you can configure your own bitmasks and use them for any kind of permissions in your scripts and applications. The implementation is relatively simple as well.

To begin, define your permissions scheme. When setting up your scheme, the easiest method is to use powers of two (the reasoning behind this will become apparent when you see the decoding function). Here's a sample permission scheme:

```
define('PERMISSION_DENIED', 0);
define('PERMISSION_READ', 1);
define('PERMISSION_ADD', 2);
define('PERMISSION_UPDATE', 4);
define('PERMISSION_DELETE', 8);
```

After you have your permissions scheme, you can apply it wherever you want. Let's use user permissions as an example. Say you wanted to create a user that had permissions to read and delete a log file. Using the above definitions, you would set the user's permission to "9".

So, how do you properly decode that permission and apply it to the user? This can be easily achieved by using the following function:

```
/**
 * Correct the variables stored in array.
 * @param integer $mask Integer of the bit
 * @return array
 */
function bitMask($mask = 0) {
    if(!is_numeric($mask)) {
        return array();
    }
    $return = array();
    while ($mask > 0) {
        for($i = 0, $n = 0; $i <= $mask; $i = 1 * pow(2, $n), $n++) {
            $end = $i;
        }
        $return[] = $end;
        $mask = $mask - $end;
    }
    sort($return);
    return $return;
}
```

What this function does is to break down the permission (\$mask - the bit mask) into its components that are powers of 2 and return them in an array. If you did a `print_r()` of the above functions return with our example of "9" you would get:

```
array(  
    0 => 1, // READ  
    1 => 8  // DELETE  
);
```

Now that you have your array of permissions, you could use the "in_array" function to check if a user has permission to perform a requested action. Take a look at this sample code:

```
// ...  
$_ARR_permission = bitMask(9);  
// ...  
if(in_array(PERMISSION_READ, $_ARR_permission)) {  
    // [...]  
}  
else {  
    echo 'Access denied.';  
}
```

That's it! A simple, elegant, often-needed solution to a common problem.