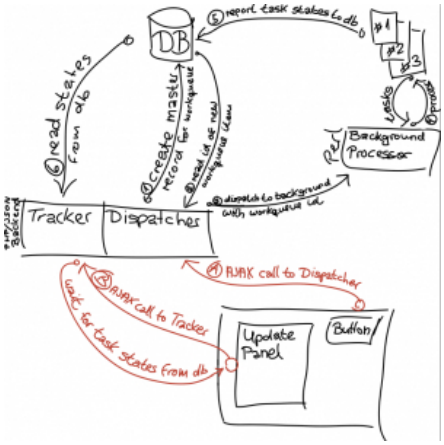
Run command in background from PHP

While working on my newest project, a PHP- and AJAX-based network management application, I came along the need to run multiple commands on the server and track their return states (basically if they were run successful or not).

While sticking together my framework, I found that forking a process into background from PHP requires some additional steps if you don't want to break AJAX functionality.

Here's a sketch of the basic concept. It involves a Dispatcher, which creates a workqueue item in a database, and then forks a background process, which will perform various tasks. The latter will then report back the states of the tasks performed to the workqueue item in the database.
On the other end, a Tracker will check the database for the running states and push progress information into the browser.



While it was working to some extent, my initial attempts at forking a background process from PHP failed terribly, because PHP always kept waiting for the command to complete.
This behaviour had the side effect that the Dispatcher couldn't immediately return the workqueue id to the AJAX client (the browser). But since the workqueue id was required by the AJAX-client to subsequently fire the Tracker, the initial request to the Dispatcher had to be synchronous instead of asynchronous.
The blocking behaviour did however screw everything, so I had to find a way around.

Of course, I was thinking about the various methods in calling external commands from PHP, starting with simple backticks, exec(), passthru() and even popen(), despite some others as well.

People familiar to Unix/Linux environments would of course say, that a background task can be started easily by appending an ampersand to the end, making it look like this:

```
exec( '/path/to/my/command &' );
system( '/path/to/my/command &' );
```

But that does not work in PHP, because all calls are always waiting for the command to return.
PHP effectively binds the STDERR and STDOUT I/O streams during execution, which is why a simple ampersand doesn't work out.

To get this to work, one must really detach the to-be-backgrounded program from the controlling terminal to trick PHP into

returning from exec.

This can be done easily by redirecting STDOUT and STDERR to a logfile or /dev/null before background the program, and just return the pid.
That would then look similar to this:

```
exec( '/path/to/some/program > /dev/null 2>&1 & echo $!' );
```

The above example would detach the programm into background while returning control to your PHP application at once.

Consider the output destination being required in any case, otherwise the I/O streams won't get detached.
So this syntax, while being perfectly valid, will definitely run your command, but won't place it into background at all:

```
exec( '/path/to/some/program 2>&1 & echo $!' );
```

That's it :-)