

**NAME**

**setup\_log** sets up the log file

**SYNOPSIS**

```
setup_log
  path=$0
  [log=file|both|stdout|null [file]]
  [keep=<number> [30]]
```

**DESCRIPTION**

**setup\_log** function sets up the log file. It aims at two things, one is simplicity:

In your shell program, you simply write:

```
setup_log path=$0 [other options]
```

That is it. Logging are taken care of. You can go ahead and do whatever you need to do. This eliminate the need to write basically the same code over and over again in every shell program to handle the log files.

The other purpose is it enforces certain practices (best or not). (Like in the case of LaTeX versus TeX, you trade the convenience with some flexibility.) The most noticeable one is the log file is located and formatted as:

```
<prog_dir>/log/<prog_base>/<prog_base>.log.YYYY-MM-DD_HH:MM:SS
```

The location of the log file is not a freely adjustable parameter, it is derived from the location of the program. This is done because you often need to cross check the program and its log files. And the fact the program and log files share are under the same parent directory allows an easy configuration of browser to see both the program and logs. (Please see RATIONALE section for other considerations.)

“path=\$0” is the fixed, mandatory option. This is because once inside the shell, you do not know the calling program name any more in ksh93.

“log=file|both|stdout|null” is optional, which specifies where you want the output to go:

```
"file"   = log file, this is the default.
"both"   = both the log file and standard out (screen)
"stdout" = the function does nothing
"null"   = stdout and stderr go to /dev/null
```

“keep=<number>” is optional, which specifies how many days the log files should be kept. The default is 30.

**RATIONALE**

The following factors prompted me to write this function:

- I found myself write the same code over and over again.
- I had the experience that I could not finding the log files; the previous log files were overwritten by next runs; log files are never cleaned up.
- I have seen:

```
command: foo
stdout:  /tmp/foo.out.$$
stderr:  /tmp/foo.err.$$
```

in autosys/cron, and worried what happens when /tmp is filled up; what happens when server reboots; and when command foo is run manually or via other means, where the log files go.

- I have seen:

```
command: foo
stdout: /tmp/foo.out
stderr: /tmp/foo.err
```

and worried where to find the previous log files if I need them.

– I have seen:

```
command_foo >$logfile 2>&1
```

and worried that the error messages would be lost if errors occur where the output is not directed.

– I have seen the question how to direct the output to both log file and screen.

### **CAVEATS**

The function uses exit trap. It will be in conflict with your program, or other functions (such as `lock_unlock`) which also use exit trap.

There are simple solutions, which is to append the trap actions, instead of replacing previous one. This is discussed:

```
http://www.unixreview.com/documents/s=9040/ur0402g/
```

but I have not figured it out how to make it generic without making things more complicated.

### **AUTHOR**

Michael Wang <[xw73@columbia.edu](mailto:xw73@columbia.edu)>.

### **VERSION HISTORY**

version 1.0, 2004-04-27, [xw73@columbia.edu](mailto:xw73@columbia.edu).  
IPO.