如何开发标准的 Shell Logger 日志程序

导读：

在 Linux/Unix 下开发后台程序，经常会用到 Shell 脚本，如何输出类似于 log4j、log4cpp 和 log4go 这样的标准化日志？本文尝试编写一组适用于 Shell 的标准日志函数。

一、前言

日志的英文单词是 log，其本来的意义是原木。在远古时代没有文字，没有纸和笔，原始人在原木上划一道印记，代表发生了一件事，或者是度过了一天。后来，有了书写工具和记录载体，语言和文字也发展了，能记录事件发生的时间、地点、人物和事件经过，于是有了古代人类历史。西方最早的历史书是古希腊时期的《伯罗奔尼撒战争史》，中国先秦时期的历史典籍有《春秋》、《战国策》和《史记》。

历史书是人类文明发展进步的记载，而如果把程序也看做有生命的事物，那么程序运行时产生的日志也是程序生命历程的痕迹。

标准化的日志有几个共同特点：日期时间格式标准化、日志消息级别标准化、进程 PID 或者程序名标准化以及消息正文格式化。

二、程序日志的功用

通过程序日志，我们能了解或者做到：

了解程序的启动、运行、交互和结束。程序的启动和结束，应当分别输出至少一条日志记录。

了解运行时日志记录程序执行时的做了什么，发生了什么故障。对于设计良好的软件，其日志输出也应该工整、规范，内容详尽，记录清晰。了解发生了什么，没有发生什么。发生过什么会有记录。预期的事件没有相应日志记录，那就是没有发生。

了解运行时的程序错误或者数据错误。开启了调试级别（Debug）的日志输出，能反映错误发生的详细过程。有经验的排障工程师或者程序员，从日志记录的细微之处，分析程序

错误发生的根源，定位到引起错误的源代码文件、函数，甚至代码行。

进行程序性能分析。日志记录的时间戳，能精确到时分秒，甚至毫秒和微秒。任意两条日志记录的时间差，反映了两个时间点之间程序执行的时长。时间差长的，执行耗时也长，性能优化时应优先考虑、安排。

三、分级日志及其功能

程序日志的分级控制，可以说是日志的灵魂。程序在调试阶段和运行阶段，共用同一套源代码：在调试阶段，希望输出更多的日志记录，便于调试找出程序错误，而在运行阶段，希望输出较少的日志记录，因为太多太详细的日志输出会消耗资源、影响程序性能，有时低级别太多会淹没高级别日志。多和少之间很难两全，而日志分级能兼顾多和少，各取所需、两全其美。

通过设置日志级别参数，低于该级别的日志不会生成，也不会输出，减少资源消耗。只有高于该级别的日志消息才会生成、输出，最大限度保留必要的信息。

分级日志的功能如下：

1、同时支持控制台和日志文件等输出方向。并且可以通过日志库函数设置为输出或者不输出。

2、支持日志消息的格式化输出，全部内容包括：消息时间、进程 PID、消息级别、消息正文内容。消息时间包含年月日和时分秒，秒后面精确到微秒级，便于估算应用的具体操作耗时。输出纳秒级时间意义不大，反而会让消息更复杂，虽说技术上可以支持，但是暂时忽略纳秒级。

3、支持日志消息的（调试）级别控制。初始化时设置消息（过滤）级别；在运行时，只有高于或等于该级别的消息才会输出，低于该级别的消息会被自动过滤不输出。上层应用可以灵活设置日志级别，控制日志消息输出的数量，例如：在调试时设置日志级别为 DEBUG，会输出 DEBUG 及以上级别的日志消息，获得更详细调试信息。而在正常运行时，设置日志级别为 INFO，将会自动忽略 DEBUG 消息，日志文件会更简洁、干净。

4、支持运行时时区设置。符合当地时区的时间消息头，能同时被中国国内和国外的用户所接受。

5、支持日志消息的语言模板管理（待实现）。同时支持多种主流语言，运行时程序预选一种语言，输出时自动套用语言模板，自动填充变量，输出语言定制化的日志消息。

四、分级日志源代码

作者曾经写过几个语言版本的日志程序 Logger，比如 Java、C/C++、MT5、xScript 等语言。所以写 Shell 脚本的 Logger 程序可谓轻车熟路，没有遇到困难。大约花一小时写完代码，再花一小时调试，就基本成型了。因为 date 命令对时区环境变量 TZ 敏感，所以 Shell 版的 Logger 程序额外增加了时区功能。设置 TZ 时区变量，可以很容易地把日志时间戳，输出为相应时区的时间。例如北京位于东半球的东八区，洛杉矶位于西半球的西八区，属于美国太平洋标准时区。

主要函数清单如下：

1、函数 logger_set_timezone()

设置日志消息的时区。

2、函数 logger_set_output()

设置日志消息的输出文件路径。

3、函数 logger_set_lang()

设置日志消息的语言。

4、函数 logger_debug()

输出调试级别的日志消息。

5、函数 logger_info()

输出信息级别的日志消息。

6、函数 logger_warn()

输出警告级别的日志消息。

7、函数 logger_error()

输出错误级别的日志消息。

8、函数 logger_alert()

输出报警级别的日志消息。

9、函数 logger_fatal()

输出致命级别的日志消息。

日志函数库的源代码如下：

```
#!/usr/bin/bash


## logger.sh
## Define constants
LOG_LANG_ALL="zh en fr"
LOG_MSG_TEMPLATE_AFFIX="diag-msg_"
LOG_OUTPUT_FILE_DEFAULT="../logs/diag.log"


## Define variables
LOG_DEBUG_LEVEL=0
LOG_OUTPUT_FILE=$LOG_OUTPUT_FILE_DEFAULT
LOG_OUTPUT_CONSOLE=true
LOG_LANG=en
LOG_MSG_TEMPLATE="diag-msg_${LOG_LANG}.tmpl"
TZ="Asia/Shanghai"


## Set time zone of message
logger_set_timezone()
{
    if [ $# -ge 1 ]; then
```

```
        TZ=$1
        logger_info "Set timezone, TZ=${TZ}."
    fi
}

## Set logfile path to output
logger_set_output()
{
    if [ $# -ge 1 ]; then
        LOG_OUTPUT_FILE=$1
    fi
}

## Set language
logger_set_lang()
{
    if [ $# -le 0 ]; then
        return;
    fi
    for lang in ${LOG_LANG_ALL}; do
        if [ ${lang} = $1 ]; then
            LOG_LANG=${lang}

LOG_MSG_TEMPLATE="${LOG_MSG_TEMPLATE_AFFIX}${LOG_LANG}.tmpl"
        fi
    done
}

## Logger output
logger_output_private()
```

```
{
    if [ $# -le 0 ]; then
        return;
    fi
    info_level=$1
    msg_id=$2
    shift
    millisecond=$(expr $(date +%N) / 1000)
    debug_msg_head="$(TZ=$TZ date '+%Y-%m-%d %H:%M:%S').$(printf '%06d' ${millisecond}) [$$] - ${info_level} "
    if [ "${msg_id:0:3}" = "MSG" ]; then
        shift
        pattern=$($msg_id)
        debug_msg="${debug_msg_head}""$(printf $pattern $@ )"
    else
        debug_msg="${debug_msg_head}$@"
    fi
    #debug_msg=${debug_msg_head}$(printf $2 )
    echo "${debug_msg}" >> ${LOG_OUTPUT_FILE}
    if [ ${LOG_OUTPUT_CONSOLE} = true ]; then
        echo "${debug_msg}"
    fi
}

## Logger debug 0
logger_debug()
{
    if [ ${LOG_DEBUG_LEVEL} -le 0 ]; then
        logger_output_private "DEBUG" $@
    fi
```

```
}

## Logger info 1
logger_info()
{
	if [ ${LOG_DEBUG_LEVEL} -le 1 ]; then
		logger_output_private "INFO" $@
	fi
}

## Logger warn 2
logger_warn()
{
	if [ ${LOG_DEBUG_LEVEL} -le 2 ]; then
		logger_output_private "WARN" $@
	fi
}

## Logger warn 3
logger_error()
{
	if [ ${LOG_DEBUG_LEVEL} -le 3 ]; then
		logger_output_private "ERROR" $@
	fi
}

## Logger alert 4
logger_alert()
{
```

```
    if [ ${LOG_DEBUG_LEVEL} -le 4 ]; then
        logger_output_private "ALERT" $@
    fi
}

## Logger fatal 5
logger_fatal()
{
    if [ ${LOG_DEBUG_LEVEL} -le 5 ]; then
        logger_output_private "FATAL" $@
    fi
}
```

五、测试日志程序

编写测试用 Shell 脚本，调用日志函数库，设置语言、时区，分别输出调试、信息、警告、错误、报警和致命级别的日志消息。先设置语言和时区为英文和美国洛杉矶时间，然后设置为中文和亚洲上海时间，先后测试两遍。

日志功能测试代码：

```
#!/usr/bin/bash
#test-common-logger.sh
## include shells
. ./common/logger.sh

logger_info    "TEST: logger"

## Set config
```

```
logger_set_lang "en";
logger_set_timezone 'America/Los_Angeles'


## Test
logger_debug "This is DEBUG level."
logger_info    "This is INFO level."
logger_warn    "This is WARN level."
logger_alert "This is ALERT level."
logger_fatal "This is FATAL level."
logger_fatal ""


## Set config
logger_set_lang "zh";
logger_set_timezone 'Asia/Shanghai'


## Test
logger_debug "这是 调试 级别."
logger_info    "这是 信息 级别."
logger_warn    "这是 警告 级别."
logger_alert "这是 警报 级别."
logger_fatal "这是 致命 级别."
logger_fatal ""
```

执行脚本 test-common-logger.sh，测试日志函数库。期望的日志输出如下：

```
[root@dev bin]# ./test/test-common-logger.sh
2019-08-29 01:36:45.115048 [5024] - INFO TEST: logger
2019-08-28 10:36:45.121267 [5024] - INFO Set timezone, TZ=America/Los_Angeles.
2019-08-28 10:36:45.142414 [5024] - DEBUG This is DEBUG level.
2019-08-28 10:36:45.146084 [5024] - INFO This is INFO level.
```

```
2019-08-28 10:36:45.149772 [5024] - WARN This is WARN level.
2019-08-28 10:36:45.153572 [5024] - ALERT This is ALERT level.
2019-08-28 10:36:45.157412 [5024] - FATAL This is FATAL level.
2019-08-28 10:36:45.161269 [5024] - FATAL
2019-08-29 01:36:45.165227 [5024] - INFO Set timezone, TZ=Asia/Shanghai.
2019-08-29 01:36:45.169012 [5024] - DEBUG 这是 调试 级别.
2019-08-29 01:36:45.172777 [5024] - INFO 这是 信息 级别.
2019-08-29 01:36:45.176555 [5024] - WARN 这是 警告 级别.
2019-08-29 01:36:45.180245 [5024] - ALERT 这是 警报 级别.
2019-08-29 01:36:45.183894 [5024] - FATAL 这是 致命 级别.
2019-08-29 01:36:45.187643 [5024] - FATAL
```

六、工作小结

本文介绍了如何编写一个标准化的 Shell 日志函数库，读者可以下载 logger 函数库脚本，然后在自己编写的脚本内引用，输出清晰、美观、符合公认标准的日志消息。

目前这个日志函数库已经支持设置日志消息的时区，对中国的用户可以使用东八区时间，也可以使用默认的格林尼治标准时间。当然全球各地的用户也可以设置时区为当地时间。

七、下一步工作

日志函数库支持使用任何一种计算机能表达的语言文字输出日志内容。

如果一个程序要同时支持输出多种文字的日志消息内容，日志消息的模板化是比较好的解决办法。普通的消息正文由固定模板和嵌入固定模板的可变部分组成，使用类似于 printf 函数，能实现日志内容格式化输出。每支持一种语言，都要编辑一套相应语言的消息模板。

下一步的工作是根据代码设置的语言，自动适配相应语言的消息模板，达到固定模板与可变部分在开发时解耦、在运行时耦合的目的。

八、源码下载

源代码托管在 github.com 源码仓库，源代码随时可能会更新。

用下面的命令可以直接克隆源码：

git clone https://github.com/solomonxu/docker-fast-build-shell.git

Cantact to the author:

Email: solomonxu@163.com

WeChat: solomonxu9999