Arduino Syslog Client Library

Aus DL8RDS Wiki

Inhaltsverzeichnis

Project Definition

We intend to upgrade our ATV relay DB0MHB in a way that it logs every event right on the second over the network to a syslog server. Since we are going to do this with an Arduino system, I understood that there is no such thing as a Syslog client library for Arduino.

If something does not exist, you need to change that.

Please visit my Google Code site:

- http://code.google.com/p/ardusyslog/

Advance Tests

Understand the format of a Syslog message

The format of a syslog message is defined in RFC5424. It is depicted as a transformational grammar:

```
    SYSLOG-MSG      = HEADER SP STRUCTURED-DATA [SP MSG]
```

```
    HEADER          = PRI VERSION SP TIMESTAMP SP HOSTNAME
                      SP APP-NAME SP PROCID SP MSGID
    PRI             = "<" PRIVAL ">"
    PRIVAL          = 1*3DIGIT ; range 0 .. 191
    VERSION         = NONZERO-DIGIT 0*2DIGIT
    HOSTNAME        = NILVALUE / 1*255PRINTUSASCII
```

```
    APP-NAME        = NILVALUE / 1*48PRINTUSASCII
    PROCID          = NILVALUE / 1*128PRINTUSASCII
    MSGID           = NILVALUE / 1*32PRINTUSASCII
```

```
    TIMESTAMP       = NILVALUE / FULL-DATE "T" FULL-TIME
    FULL-DATE       = DATE-FULLYEAR "-" DATE-MONTH "-" DATE-MDAY
    DATE-FULLYEAR   = 4DIGIT
    DATE-MONTH      = 2DIGIT  ; 01-12
    DATE-MDAY       = 2DIGIT  ; 01-28, 01-29, 01-30, 01-31 based on
```

```
                          ; month/year
FULL-TIME       = PARTIAL-TIME TIME-OFFSET
PARTIAL-TIME    = TIME-HOUR ":" TIME-MINUTE ":" TIME-SECOND
                  [TIME-SECFRAC]
TIME-HOUR       = 2DIGIT  ; 00-23
TIME-MINUTE     = 2DIGIT  ; 00-59
TIME-SECOND     = 2DIGIT  ; 00-59
TIME-SECFRAC    = "." 1*6DIGIT
TIME-OFFSET     = "Z" / TIME-NUMOFFSET
TIME-NUMOFFSET  = ("+" / "-") TIME-HOUR ":" TIME-MINUTE
STRUCTURED-DATA = NILVALUE / 1*SD-ELEMENT
SD-ELEMENT      = "[" SD-ID *(SP SD-PARAM) "]"
SD-PARAM        = PARAM-NAME "=" %d34 PARAM-VALUE %d34
SD-ID           = SD-NAME
PARAM-NAME      = SD-NAME
PARAM-VALUE     = UTF-8-STRING ; characters '"', '\' and
                               ; ']' MUST be escaped.
SD-NAME         = 1*32PRINTUSASCII
                  ; except '=', SP, ']', %d34 (")
MSG             = MSG-ANY / MSG-UTF8
MSG-ANY         = *OCTET ; not starting with BOM
MSG-UTF8        = BOM UTF-8-STRING
BOM             = %xEF.BB.BF
UTF-8-STRING    = *OCTET ; UTF-8 string as specified
                  ; in RFC 3629
OCTET           = %d00-255
SP              = %d32
PRINTUSASCII    = %d33-126
NONZERO-DIGIT   = %d49-57
DIGIT           = %d48 / NONZERO-DIGIT
NILVALUE        = "-"
```

This grammar defines nonstructured as well as structured presentations. So very basically, the Syslog message consists of the following basic form:

```
<PRI> TIMESTAMP TAG MESSAGE
```

The PRI value is an integer number which calculates by the following metric:

```
8 x (facility code) + (severity code)
```

where the individual codes are those:

Facilities:

```
          0             kernel messages
          1             user-level messages
          2             mail system
          3             system daemons
          4             security/authorization messages
          5             messages generated internally by syslogd
          6             line printer subsystem
          7             network news subsystem
          8             UUCP subsystem
          9             clock daemon
         10             security/authorization messages
         11             FTP daemon
         12             NTP subsystem
         13             log audit
         14             log alert
         15             clock daemon (note 2)
         16             local use 0  (local0)
         17             local use 1  (local1)
         18             local use 2  (local2)
         19             local use 3  (local3)
         20             local use 4  (local4)
         21             local use 5  (local5)
         22             local use 6  (local6)
         23             local use 7  (local7)
```

Severities:

```
           0       Emergency: system is unusable
           1       Alert: action must be taken immediately
           2       Critical: critical conditions
           3       Error: error conditions
           4       Warning: warning conditions
           5       Notice: normal but significant condition
           6       Informational: informational messages
           7       Debug: debug-level messages
```

The TIMESTAMP may be the NILVALUE if there is no time available.

Enable my local Syslog daemon to receive log messages over the network

Running a NATTY Ubuntu, I noticed that modern Ubuntu distros use Rainer Gerhards' **rsyslog** implementation.

It has a section in the config file /etc/rsyslog.conf which just needs to be uncommented:

```
# provides UDP syslog reception
$ModLoad imudp
$UDPServerRun 514
```

Upon restarting the rsyslog service with the command **service rsyslog restart** you can check with **netstat -tulpen** if the service is listening on UDP port 514.

Trace the logger command

The **logger** command is a nice little commandline tool that permits local services and scripts to log to the system's syslog. It is not able to send messages over the network, but uses /dev/log instead. Since there are no network packages we can capture, we need to use **strace** to capture everything the logger is doing:

```
markus@note:~$ strace logger -p local0.warn -t test huhuasder345345sdfsdfhsdfee
execve("/usr/bin/logger", ["logger", "-p", "local0.warn", "-t", "test", "huhuasder345345sdfsdfhsdfee"], [/* 41 vars */]) =
brk(0)                                  = 0x10f3000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f22b36b5000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY)      = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=152165, ...}) = 0
mmap(NULL, 152165, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f22b368f000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\20\360\1\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=1638120, ...}) = 0
mmap(NULL, 3749080, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f22b3103000
mprotect(0x7f22b328d000, 2093056, PROT_NONE) = 0
mmap(0x7f22b348c000, 20480, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x189000) = 0x7f22b348c000
mmap(0x7f22b3491000, 21720, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f22b3491000
close(3)                                = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f22b368e000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f22b368c000
arch_prctl(ARCH_SET_FS, 0x7f22b368c720) = 0
mprotect(0x7f22b348c000, 16384, PROT_READ) = 0
mprotect(0x601000, 4096, PROT_READ)     = 0
mprotect(0x7f22b36b7000, 4096, PROT_READ) = 0
munmap(0x7f22b368f000, 152165)          = 0
brk(0)                                  = 0x10f3000
brk(0x1114000)                          = 0x1114000
open("/usr/lib/locale/locale-archive", O_RDONLY) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=8310192, ...}) = 0
mmap(NULL, 8310192, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f22b2916000
close(3)                                = 0
```

```
close(1)                                = 0
open("/etc/localtime", O_RDONLY)        = 1
fstat(1, {st_mode=S_IFREG|0644, st_size=2309, ...}) = 0
fstat(1, {st_mode=S_IFREG|0644, st_size=2309, ...}) = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f22b36b4000
read(1, "TZif2\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\10\0\0\0\10\0\0\0\0"..., 4096) = 2309
lseek(1, -1467, SEEK_CUR)               = 842
read(1, "TZif2\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\t\0\0\0\t\0\0\0\0"..., 4096) = 1467
close(1)                                = 0
munmap(0x7f22b36b4000, 4096)            = 0
socket(PF_FILE, SOCK_DGRAM|SOCK_CLOEXEC, 0) = 1
connect(1, {sa_family=AF_FILE, path="/dev/log"}, 110) = 0
sendto(1, "<132>Jun 23 20:46:53 test: huhua"..., 54, MSG_NOSIGNAL, NULL, 0) = 54
close(1)                                = 0
exit_group(0)                           = ?
```

Note the last four lines: The tool opens the /dev/log device and the following string is being sent to it:

```
<132>Jun 23 20:46:53 test: huhua (......)
```

Manually Sending a Syslog Message

Since we now know how a syslog message looks like, there is the assumption the the syslog server will just accept the message when we submit it via the network. We can do that with the Swiss Army Knife of network engineering, the tool **netcat**, also known as **nc**:

```
markus@note:~$ nc -u localhost 514
<132>Jun 23 20:46:53 test: huhua
```

and it just comes out in the /var/log/syslog file as expected:

```
Jun 23 20:46:53 localhost test: huhua
```

So we have the proof that a syslog client just needs to work accordingly.

Implementation and Tests

Since we have seen how Syslog works across the network, we will now implement the client library.

Von „http://www.dl8rds.de/index.php?title=Arduino_Syslog_Client_Library&oldid=1848“

- Diese Seite wurde zuletzt am 23. Juni 2011 um 22:56 Uhr geändert.