

The Alog Toolbox Command Line Utilities

June 2018

Michael Benjamin, mikerb@mit.edu
Department of Mechanical Engineering, CSAIL
MIT, Cambridge MA 02139

1	Overview	2
2	An Example .alog File	3
3	The alogscan Tool	3
3.1	Command Line Usage for the alogscan Tool	3
3.2	Example Output from the alogscan Tool	4
4	The alogclip Tool	6
4.1	Command Line Usage for the alogclip Tool	6
4.2	Example Output from the alogclip Tool	7
5	The aloggrep Tool	7
5.1	Command Line Usage for the aloggrep Tool	8
5.2	Example Output from the aloggrep Tool	8
6	The alogrm Tool	9
6.1	Command Line Usage for the alogrm Tool	9
6.2	Example Output from the alogrm Tool	9
7	The aloghelm Tool	10
7.1	The Life Events (-life) Option in the aloghelm Tool	10
7.2	The Modes (-modes) Option in the aloghelm Tool	11
7.3	The Behaviors Option in the aloghelm Tool	13
7.4	Command Line Usage for the aloghelm Tool	15
8	The alogiter Tool	16
8.1	Command Line Usage for the alogiter Tool	16
8.2	Example Output from the alogiter Tool	16
9	The alogsplit Tool	18
9.1	Naming and Cleaning the Auto-Generated Split Directories	18
9.2	Command Line Usage for the alogsplit Tool	18
9.3	Example Output from the alogsplit Tool	19
10	The alogpare Tool	20
10.1	Mark Variables Define Events of Interest	20
10.2	The Pare List of Variables to be Pared	20
10.3	The Hit List of Variables to be Removed Completely	20
10.4	Command Line Usage for the alogpare Tool	21
10.5	Planned additions to the alogpare Utility	21
11	The alogcd Tool	22
11.1	Producing a Time-Stamped file of Collisions and Near Misses	22

11.2 The Terse Output Option	23
11.3 Command Line Return Values	23
11.4 Command Line Usage for the alogcd Tool	23
11.5 Planned additions to the alogcd Utility	24
12 The alogcat Tool	25
12.1 Producing a Time-Stamped file of Collisions and Near Misses	25
12.2 The Terse Output Option	26
12.3 Command Line Return Values	26
12.4 Command Line Usage for the alogcd Tool	26
12.5 Planned additions to the alogcd Utility	27

1 Overview

The Alog Toolbox, in addition to the `alogview` GUI based utility, also contains a set of command line post-mission analysis utility applications:

- `alogclip`
- `aloggrep`
- `alogrm`
- `alogscan`
- `aloghelm`
- `alogiter`
- `alogsplit`
- `alogpare`
- `alogcd`
- `alogcat`

Each application manipulates or renderings `.alog` files generated by the `pLogger` application. Four of the applications, `alogclip`, `aloggrep`, `alogpare`, and `alogrm` are command-line tools for filtering a given `.alog` file to a reduced size. Reduction of a log file size may facilitate the time to load a file in a post-processing application, may facilitate its transmission over slow transmission links when analyzing data between remote users, or may simply ease in the storing and back-up procedures. The `alogscan` tool provides statistics on a given `.alog` file that may indicate how to best reduce file size by eliminating variable entries not used in post-processing. It also generates other information that may be handy in debugging a mission. The `alogsplit` tool will split a single alog file into a folder containing a dedicated alog file for each logged variable. This operation is also done automatically upon the launch of `appalogview` when launched on an alog file for the first time. The `alogview` tool is a GUI-based tool that accepts one or more `.alog` files and renders a vehicle positions over time on an operation area, provides time-correlated plots of any logged numerical MOOS variables, and renders helm autonomy mode data with plots of generated objective functions.

2 An Example .alog File

The `.alog` file used in the examples below was generated from the Alpha example mission. This file, `alpha.alog`, is found in the missions distributed with the MOOS-IvP tree. The `alpha.alog` file was created by simply running the mission as described, and can be found in:

```
moos-ivp/trunk/ivp/missions/alpha/alpha.alog
```

3 The alogscan Tool

The `alogscan` tool is a command-line application for providing statistics relating to a given `.alog` file. It reports, for each unique MOOS variable in the log file, (a) the number of lines in which the variable appears, i.e., the number of times the variable was posted by a MOOS application, (b) the total number of characters comprising the variable value for all entries of a variable, (c) the timestamp of the first recorded posting of the variable, (d) the timestamp of the last recorded posting of the variable, (e) the list of MOOS applications the posted the variable.

3.1 Command Line Usage for the alogscan Tool

The `alogscan` tool is run from the command line with a given `.alog` file and a number of options. The usage options are listed when the tool is launched with the `-h` switch:

```
$ alogscan --help or -h
```

```
1 Usage:
2   alogscan file.alog [OPTIONS]
3
4 Synopsis:
5   Generate a report on the contents of a given
6   MOOS .alog file.
7
8 Options:
9   --sort=type   Sort by one of SIX criteria:
10                 start: sort by first post of a var
11                 stop:  sort by last post of a var
12   (Default) vars: sort by variable name
13                 proc:  sort by process/source name
14                 chars: sort by total chars for a var
15                 lines: sort by total lines for a var
16
17 --appstat      Output application statistics
18 -r,--reverse  Reverse the sorting output
19 -n,--nocolors Turn off process/source color coding
20 -h,--help     Displays this help message
21 -v,--version  Displays the current release version
22 --rate_only   Only report the data rate
23 --noaux       Ignore auxilliary source info
24
25 See also: aloggrp, alogrm, alogclip, alogview
```

The order of the arguments passed to `alogscan` do not matter. The lines of output are sorted by grouping variables posted by the same MOOS process or source. The sorting criteria can instead be done by alphabetical order on the variable name (`--sort=vars`), the total characters in the file due to a variable (`--sort=chars`), the total lines in the file due to a variable (`--sort=lines`), the time of the first posting of the variable (`--sort=start`), or the time of the last posting of the variable (`--sort=stop`). The order of the output may be reversed (`-r`, `--reverse`). By default, the entries are color-coded by the variable source, using the few available terminal colors (there are not many). When unique colors are exhausted, the color reverts back to the default terminal color in effect at the time.

3.2 Example Output from the alogscan Tool

The output shown below was generated from the `alpha.log` file generated by the Alpha example mission.

```
$ alogscan file.alog
```

Variable Name	Lines	Chars	Start	Stop	Sources
DB_CLIENTS	282	22252	-0.38	566.42	MOOSDB_alpha
DB_TIME	556	7132	1.21	566.18	MOOSDB_alpha
DB_UPTIME	556	7173	1.21	566.18	MOOSDB_alpha
USIMMARINE_STATUS	276	92705	0.39	565.82	uSimMarine
NAV_DEPTH	6011	6011	1.43	566.38	uSimMarine
NAV_HEADING	6011	75312	1.43	566.38	uSimMarine
NAV_LAT	6011	74799	1.43	566.38	uSimMarine
NAV_LONG	6011	80377	1.43	566.38	uSimMarine
NAV_SPEED	6011	8352	1.43	566.38	uSimMarine
NAV_STATE	6011	18033	1.43	566.38	uSimMarine
NAV_X	6011	72244	1.43	566.38	uSimMarine
NAV_Y	6011	77568	1.43	566.38	uSimMarine
NAV_YAW	6011	80273	1.43	566.38	uSimMarine
BHV_IPF	2009	564165	46.26	542.85	pHelmIvP
CREATE_CPU	2108	2348	46.26	566.33	pHelmIvP
CYCLE_INDEX	5	5	44.98	543.09	pHelmIvP
DEPLOY	3	14	3.84	543.09	pHelmIvP,pMarineViewer
DESIRED_HEADING	2017	5445	3.85	543.09	pHelmIvP
DESIRED_SPEED	2017	2017	3.85	543.09	pHelmIvP
HELM_IPF_COUNT	2108	2108	46.26	566.32	pHelmIvP
HSLINE	1	3	3.84	3.84	pHelmIvP
IVPHELM_DOMAIN	1	29	3.84	3.84	pHelmIvP
IVPHELM_ENGAGED	462	3342	3.85	566.32	pHelmIvP
IVPHELM_MODESET	1	0	3.84	3.84	pHelmIvP
IVPHELM_POSTINGS	2014	236320	46.26	543.33	pHelmIvP
IVPHELM_STATEVARS	1	20	44.98	44.98	pHelmIvP
IVPHELM_SUMMARY	2113	612685	44.98	566.33	pHelmIvP
LOOP_CPU	2108	2348	46.26	566.33	pHelmIvP
PC_hsline	1	9	44.98	44.98	pHelmIvP
PC_waypt_return	3	14	44.98	543.33	pHelmIvP
PC_waypt_survey	3	14	44.98	543.33	pHelmIvP
PHELMIVP_STATUS	255	198957	3.85	565.12	pHelmIvP
PLOGGER_CMD	1	17	3.84	3.84	pHelmIvP
PWT_BHV_HSLINE	1	1	44.98	44.98	pHelmIvP
PWT_BHV_WAYPT_RETURN	3	5	44.98	543.09	pHelmIvP
PWT_BHV_WAYPT_SURVEY	2	4	44.98	462.90	pHelmIvP
RETURN	4	19	3.84	543.09	pHelmIvP,pMarineViewer
STATE_BHV_HSLINE	1	1	44.98	44.98	pHelmIvP
STATE_BHV_WAYPT_RETURN	4	4	44.98	543.33	pHelmIvP
STATE_BHV_WAYPT_SURVEY	3	3	44.98	463.15	pHelmIvP
SURVEY_INDEX	10	10	44.98	429.70	pHelmIvP
SURVEY_STATUS	1116	77929	45.97	462.90	pHelmIvP
VIEW_POINT	4034	101662	44.98	543.33	pHelmIvP
VIEW_SEGLIST	4	273	44.98	543.33	pHelmIvP
WPT_INDEX	1	1	463.15	463.15	pHelmIvP
WPT_STAT	223	15626	463.15	543.09	pHelmIvP
LOGGER_DIRECTORY	56	1792	1.07	559.19	pLogger
PLOGGER_STATUS	263	331114	1.07	566.40	pLogger
DESIRED_RUDDER	10185	150449	-9.28	545.18	pMarinePID
DESIRED_THRUST	10637	20774	-9.28	566.52	pMarinePID
MOOS_DEBUG	5	39	-9.31	545.23	pMarinePID,pHelmIvP
PMARINEPID_STATUS	279	81990	0.95	566.28	pMarinePID
HELM_MAP_CLEAR	1	1	-1.56	-1.56	pMarineViewer
MOOS_MANUAL_OVERRIDE	1	5	44.65	44.65	pMarineViewer
PMARINEVIEWER_STATUS	270	95560	-0.95	564.89	pMarineViewer
NODE_REPORT_LOCAL	1159	207535	1.15	565.91	pNodeReporter
PNODEREPORTER_STATUS	233	50534	-0.37	563.93	pNodeReporter

Total variables: 57					
Start/Stop Time: -9.31 / 566.52					

When the `-appstat` command line option is included, a second report is generated, after the above report, that provides statistics keyed by application, rather than by variable. For each application that has posted a variable recorded in the given `.alog` file, the number of lines and characters are recorded, as well as the percentage of total lines and characters. An example of this report:

MOOS Application	Total Lines	Total Chars	Lines/Total	Chars/Total
MOOSDB_alpha	1394	36557	1.37	1.08
uSimMarine	54375	585674	53.57	17.29
pHelmIvP	22642	1825437	22.31	53.89
pLogger	319	332906	0.31	9.83
pMarinePID	21106	253252	20.80	7.48
pMarineViewer	279	95599	0.27	2.82
pNodeReporter	1392	258069	1.37	7.62

Further Tips

- If a small number of variables are responsible for a relatively large portion of the file size, and are expendable in terms of how data is being analyzed, the variables may be removed to ease the handling, transmission, or storage of the data. To remove variables from existing files, the `alogrm` tool described in Section 6 may be used. To remove the variable from future files, the `pLogger` configuration may be edited by either removing the variable from the list of variables explicitly requested for logging, or if `WildcardLogging` is used, mask out the variable with the `WildcardOmitPattern` parameter setting. See the `pLogger` documentation.
- The output of `alogscan` can be further distilled using common tools such as `grep`. For example, if one only wants a report on variables published by the `pHelmIvP` application, one could type:

```
$ alogscan alpha.alog | grep pHelmIvP
```

4 The alogclip Tool

The `alogclip` tool will prune a given `.alog` file based on a given beginning and end timestamp. This is particularly useful when a log file contains a sizeable stretch of data logged after mission completion, such as data being recorded while the vehicle is being recovered or sitting idle topside after recovery.

4.1 Command Line Usage for the alogclip Tool

The `alogclip` tool is run from the command line with a given `.alog` file, a start time, end time, and the name of a new `.alog` file. By default, if the named output file exists, the user will be prompted before overwriting it. The user prompt can be bypassed with the `-f, --force` option. The usage options are listed when the tool is launched with the `-h` switch:

```
$ alogclip --help or -h
```

Usage:
alogclip in.alog mintime maxtime [out.alog] [OPTIONS]

Synopsis:
Create a new MOOS .alog file from a given .alog file
by removing entries outside a given time window.

Standard Arguments:
in.alog - The input logfile.
mintime - Log entries with timestamps below mintime
will be excluded from the output file.
maxtime - Log entries with timestamps above mintime
will be excluded from the output file.
out.alog - The newly generated output logfile. If no
file provided, output goes to stdout.

Options:
-h,--help Display this usage/help message.
-v,--version Display version information.
-f,--force Overwrite an existing output file
-q,--quiet Verbose report suppressed at conclusion.

Further Notes:
(1) The order of arguments may vary. The first alog
file is treated as the input file, and the first
numerical value is treated as the mintime.
(2) Two numerical values, in order, must be given.
(3) See also: alogscan, alogrm, aloggrep, alogview

4.2 Example Output from the alogclip Tool

The output shown below was generated from the alpha.alog file generated by the Alpha example mission.

```
$ alogclip alpha.alog new.alog 50 350
```

```
Processing input file alpha.alog...

Total lines clipped:      44,988 (44.32 pct)
  Front lines clipped:    5,474
  Back lines clipped:    39,514
Total chars clipped:    4,200,260 (43.09 pct)
  Front chars clipped:   432,409
  Back chars clipped:   3,767,851
```

5 The aloggrep Tool

The [aloggrep](#) tool will prune a given .alog file by retaining lines of the original file that contain log entries for a user-specified list of MOOS variables or MOOS processes (sources). As the name implies it is motivated by the Unix `grep` command, but `grep` will return a matched line regardless of where the pattern appears in the line. Since MOOS variables also often appear in the string content

of other MOOS variables, `grep` often returns much more than one is looking for. The `aloggrep` tool will only pattern-match on the second column of data (the MOOS variable name), or the third column of data (the MOOS source), of any given entry in a given `.alog` file.

5.1 Command Line Usage for the `aloggrep` Tool

```
$ aloggrep --help or -h
```

```
Usage:
  aloggrep in.alog [VAR] [SRC] [out.alog] [OPTIONS]

Synopsis:
  Create a new MOOS .alog file by retaining only the
  given MOOS variables or sources from a given .alog file.

Standard Arguments:
  in.alog - The input logfile.
  out.alog - The newly generated output logfile. If no
             file provided, output goes to stdout.
  VAR      - The name of a MOOS variable
  SRC      - The name of a MOOS process (source)

Options:
  -h,--help      Displays this help message
  -v,--version   Displays the current release version
  -f,--force     Force overwrite of existing file
  -q,--quiet     Verbose report suppressed at conclusion

Further Notes:
  (1) The second alog is the output file. Otherwise the
      order of arguments is irrelevant.
  (2) VAR* matches any MOOS variable starting with VAR
  (3) See also: alogscan, alogrm, alogclip, alogview
```

Note that, in specifying items to be filtered out, there is no distinction made on the command line that a given item refers to a entry's variable name or an entry's source, i.e., MOOS process name.

5.2 Example Output from the `aloggrep` Tool

The output shown below was generated from the `alpha.alog` file generated by the Alpha example mission.

```
$ aloggrep alpha.alog NAV_* new.alog
```

```
Processing on file : alpha.alog
Total lines retained: 54099 (53.30%)
Total lines excluded: 47396 (46.70%)
Total chars retained: 3293774 (33.79%)
Total chars excluded: 6453494 (66.21%)
Variables retained: (9) NAV_DEPTH, NAV_HEADING, NAV_LAT, NAV_LONG,
NAV_SPEED, NAV_STATE, NAV_X, NAV_Y, NAV_YAW
```


6 The alogrm Tool

The `alogrm` tool will prune a given `.alog` file by removing lines of the original file that contain log entries for a user-specified list of MOOS variables or MOOS processes (sources). It may be fairly viewed as the complement of the `aloggrep` tool.

6.1 Command Line Usage for the alogrm Tool

```
$ alogrm --help or -h
```

```
Usage:
  alogrm in.alog [VAR] [SRC] [out.alog] [OPTIONS]

Synopsis:
  Remove the entries matching the given MOOS variables or sources
  from the given .alog file and generate a new .alog file.

Standard Arguments:
  in.alog - The input logfile.
  out.alog - The newly generated output logfile. If no
             file provided, output goes to stdout.
  VAR      - The name of a MOOS variable
  SRC      - The name of a MOOS process (source)

Options:
  -h,--help      Displays this help message
  -v,--version   Displays the current release version
  -f,--force     Force overwrite of existing file
  -q,--quiet     Verbose report suppressed at conclusion
  --nostr       Remove lines with string data values
  --nonum       Remove lines with double data values
  --clean       Remove lines that have a timestamp that is
                non-numerical or lines w/ no 4th column

Further Notes:
  (1) The second alog is the output file. Otherwise the
      order of arguments is irrelevant.
  (2) VAR* matches any MOOS variable starting with VAR
  (3) See also: alogscan, aloggrep, alogclip, alogview
```

Note that, in specifying items to be filtered out, there is no distinction made on the command line that a given item refers to a entry's variable name or an entry's source, i.e., MOOS process name.

6.2 Example Output from the alogrm Tool

The output shown below was generated from the `alpha.alog` file generated by the Alpha example mission.

```
$ alogrm alpha.alog NAV_* new.alog
```

```

Processing on file : alpha.alog
Total lines retained: 47396 (46.70%)
Total lines excluded: 54099 (53.30%)
Total chars retained: 6453494 (66.21%)
Total chars excluded: 3293774 (33.79%)
Variables retained: (48) BHV_IPF, CREATE_CPU, CYCLE_INDEX, DB_CLIENTS,
DB_TIME, DB_UPTIME, DEPLOY, DESIRED_HEADING, DESIRED_RUDDER, DESIRED_SPEED,
DESIRED_THRUST, HELM_IPF_COUNT, HELM_MAP_CLEAR, HSLINE, USIMMARINE_STATUS,
IVPHELM_DOMAIN, IVPHELM_ENGAGED, IVPHELM_MODESET, IVPHELM_POSTINGS,
IVPHELM_STATEVARS, IVPHELM_SUMMARY, LOGGER_DIRECTORY, LOOP_CPU, MOOS_DEBUG,
MOOS_MANUAL_OVERRIDE, NODE_REPORT_LOCAL, PC_hsline, PC_waypt_return,
PC_waypt_survey, PHELMIVP_STATUS, PLOGGER_CMD, PLOGGER_STATUS,
PMARINEPID_STATUS, PMARINEVIEWER_STATUS, PNODEREPORTER_STATUS,
PWT_BHV_HSLINE, PWT_BHV_WAYPT_RETURN, PWT_BHV_WAYPT_SURVEY, RETURN,
STATE_BHV_HSLINE, STATE_BHV_WAYPT_RETURN, STATE_BHV_WAYPT_SURVEY,
SURVEY_INDEX, SURVEY_STATUS, VIEW_POINT, VIEW_SEGLIST, WPT_INDEX, WPT_STAT

```

7 The aloghelm Tool

The `aloghelm` tool provides a few handy ways of looking at helm activity over the course of a given single alog file. This includes:

- *Life Events*: Using the `--life/-l` option, every spawning or death of a behavior is sorted into a list of life events. Section [7.1](#).
- *Mode Changes*: Using the `--modes/-m` option, every helm mode change is sorted into a list of chronological entries. Section [7.2](#).
- *Behavior States*: Using the `--bhvs/-b` option, every instance where a behavior changes states is recorded and sorted into a list of chronological entries. Section [7.3](#).

In each mode, the user may additionally specify one or more MOOS variables to be interleaved in the report as they occur chronologically

7.1 The Life Events (`-life`) Option in the aloghelm Tool

The *life events* option in `aloghelm` will scan the given alog file for all life events, defined by the spawning or destruction of a behavior instance. This information is posted by the helm in the `IVPHELM_LIFE_EVENT` variable. Example output is show below:

```
$ aloghelm file.alog --life
```

```

Processing on file : henry.alog
+++++++ (100,000) lines
+++++++ (200,000) lines
+++
233,736 lines total.
10 life events.

```

```

*****
*           Summary of Behavior Life Events           *
*****

```

Time	Iter	Event	Behavior	Behavior Type	Spawning Seed
41.27	1	spawn	loiter	BHV_Loiter	helm_startup
41.27	1	spawn	waypt_return	BHV_Waypoint	helm_startup
41.27	1	spawn	station-keep	BHV_StationKeep	helm_startup
316.78	995	spawn	ac_avd_gilda	BHV_AvoidCollision	name=avd_gilda # contact=gilda
369.72	1191	death	ac_avd_gilda	BHV_AvoidCollision	
482.92	1601	spawn	ac_avd_gilda	BHV_AvoidCollision	name=avd_gilda # contact=gilda
545.18	1833	death	ac_avd_gilda	BHV_AvoidCollision	
654.70	2228	spawn	ac_avd_gilda	BHV_AvoidCollision	name=avd_gilda # contact=gilda
751.87	2591	death	ac_avd_gilda	BHV_AvoidCollision	
809.85	2799	spawn	ac_avd_gilda	BHV_AvoidCollision	name=avd_gilda # contact=gilda

The actual output, by default, is color-code green for all spawnings and black for all deaths. The color-coding can be turned off with the additional command line argument `--nocolor`.

7.2 The Modes (`-modes`) Option in the `aloghelm` Tool

The `modes` option in `aloghelm` will scan the given `alog` and report all instances of a helm mode change.

```
$ aloghelm file.alog --modes
```

```
Processing on file : /Users/mikerb/henry.alog
=====
45.221 Mode: ACTIVE:LOITERING
=====
92.687 Mode: ACTIVE:STATION-KEEPING
=====
120.919 Mode: ACTIVE:LOITERING
=====
386.632 Mode: ACTIVE:RETURNING
=====
413.980 Mode: ACTIVE:LOITERING
=====
558.254 Mode: ACTIVE:RETURNING
=====
584.283 Mode: ACTIVE:LOITERING
=====
663.162 Mode: ACTIVE:STATION-KEEPING
=====
703.517 Mode: ACTIVE:LOITERING
=====
766.938 Mode: ACTIVE:RETURNING
=====
233,736 lines total.
```

Using the `--mode` option, it is sometimes helpful to augment the output to include certain other variable postings, by simply naming the MOOS variable on the command line. The variables and mode changes will be presented on the screen in their chronological order. For example:

```
$ aloghelm file.alog --modes CONTACT_RESOLVED
```

```
Processing on file : /Users/mikerb/henry.alog
=====
45.221 Mode: ACTIVE:LOITERING
=====
92.687 Mode: ACTIVE:STATION-KEEPING
=====
120.919 Mode: ACTIVE:LOITERING
373.392 CONTACT_RESOLVED pHelmIvP:1190:ac_avd_gilda GILDA
=====
386.632 Mode: ACTIVE:RETURNING
=====
413.980 Mode: ACTIVE:LOITERING
548.838 CONTACT_RESOLVED pHelmIvP:1832:ac_avd_gilda GILDA
=====
558.254 Mode: ACTIVE:RETURNING
=====
584.283 Mode: ACTIVE:LOITERING
=====
663.162 Mode: ACTIVE:STATION-KEEPING
=====
703.517 Mode: ACTIVE:LOITERING
755.509 CONTACT_RESOLVED pHelmIvP:2590:ac_avd_gilda GILDA
=====
766.938 Mode: ACTIVE:RETURNING

233,736 lines total.
```

7.3 The Behaviors Option in the aloghelm Tool

The *behaviors* option in `aloghelm` will scan the given alog file taking note of all helm iterations where there is a change to one or more of the four groups of (a) active, (b) running, (c) idle, or (d) completed behaviors. Example output is show below:

```
$ aloghelm file.alog --bhvs
```

```

Processing on file : henry.alog
=====
45.221 Mode: ACTIVE:LOITERING
-----
45.225 (1) Active: loiter
45.225 (1) Running:
45.225 (1) Idle: waypt_return,station-keep
=====
92.687 Mode: ACTIVE:STATION-KEEPING
-----
92.689 (172) Active: station-keep
92.689 (172) Running:
92.689 (172) Idle: loiter,waypt_return
=====
120.919 Mode: ACTIVE:LOITERING
-----
120.921 (274) Active: loiter
120.921 (274) Running:
120.921 (274) Idle: waypt_return,station-keep
-----
320.786 (995) Active: loiter,ac_avd_gilda
320.786 (995) Running:
320.786 (995) Idle: waypt_return,station-keep
-----
345.778 (1090) Active: loiter
345.778 (1090) Running: ac_avd_gilda
345.778 (1090) Idle: waypt_return,station-keep
-----
373.642 (1191) Active: loiter
373.642 (1191) Running:
373.642 (1191) Idle: waypt_return,station-keep
373.642 (1191) Completed: ac_avd_gilda
=====
386.632 Mode: ACTIVE:RETURNING
-----
386.636 (1238) Active: waypt_return
386.636 (1238) Running:
386.636 (1238) Idle: loiter,station-keep
386.636 (1238) Completed: ac_avd_gilda
=====

```

In some cases, there is interest in a particular behavior in the this kind of output. To make it easier to visually parse, the `--watch=BHV` option can be used to draw attention to each the particular behavior changes state. Example output is shown below. The primary difference is the `CHANGE` tag for each instance of a state change. In the terminal, such lines are also rendered in a different color.

```
$ aloghelm file.alog --bhvs --watch=loiter
```

```

Processing on file : henry.alog
=====
45.221 Mode: ACTIVE:LOITERING
-----
45.225 (1) Active: loiter CHANGE
45.225 (1) Running:
45.225 (1) Idle: waypt_return,station-keep
=====
92.687 Mode: ACTIVE:STATION-KEEPING
-----
92.689 (172) Active: station-keep
92.689 (172) Running:
92.689 (172) Idle: loiter,waypt_return CHANGE
=====
120.919 Mode: ACTIVE:LOITERING
-----
120.921 (274) Active: loiter CHANGE
120.921 (274) Running:
120.921 (274) Idle: waypt_return,station-keep
-----
320.786 (995) Active: loiter,ac_avd_gilda
320.786 (995) Running:
320.786 (995) Idle: waypt_return,station-keep
-----
345.778 (1090) Active: loiter
345.778 (1090) Running: ac_avd_gilda
345.778 (1090) Idle: waypt_return,station-keep
-----
373.642 (1191) Active: loiter
373.642 (1191) Running:
373.642 (1191) Idle: waypt_return,station-keep
373.642 (1191) Completed: ac_avd_gilda
=====
386.632 Mode: ACTIVE:RETURNING
-----
386.636 (1238) Active: waypt_return
386.636 (1238) Running:
386.636 (1238) Idle: loiter,station-keep CHANGE
386.636 (1238) Completed: ac_avd_gilda
=====

```

7.4 Command Line Usage for the aloghelm Tool

```
$ aloghelm --help or -h
```

Listing 7.1: Command line usage for the aloghelm tool.

```

1 Usage:
2   aloghelm file.alog [OPTIONS] [MOOSVARS]
3
4 Synopsis:
5   Perform one of several optional helm reports based on
6   helm output logged in the given .alog file.
7
8 Options:

```

```

 9  -h,--help      Displays this help message
10  -v,--version  Displays the current release version
11  -l,--life     Show report on IvP Helm Life Events
12  -b,--bhvs    Show helm behavior state changes
13  -m,--modes   Show helm mode changes
14  --watch=bhv  Watch a particular behavior for state change
15  --nocolor    Turn off use of color coding
16  --notrunc    Don't truncate MOOSVAR output (on by default)
17
18  Further Notes:
19  (1) The order of arguments is irrelevant.
20  (2) Only the first specified .alog file is reported on.
21  (3) Arguments that are not one of the above options or an
22      alog file, are interpreted as MOOS variables on which
23      to report as encountered.

```

8 The alogiter Tool

The `alogiter` tool will analyze the `ITER_GAP` and `ITER_LEN` information produced by any appcasting MOOS app. These variables indicate the ability of an application to keep up with the requested apptick frequency. For example `PHELMIVP_ITER_GAP` will be close to 1.0 when configured with an apptick of 4, and the observed apptick is also 4. The gap value will be around 2 if the observed apptick is around 2. The `PHELMIVP_ITER_LEN` is the elapsed time between the start and end of the helm iterate loop.

8.1 Command Line Usage for the alogiter Tool

```
$ alogiter --help or -h
```

Listing 8.2: Command line usage for the `alogiter` tool.

```

 1  $ alogrm -h
 2
 3  Usage:
 4  alogiter in.alog [OPTIONS]
 5
 6  Synopsis:
 7  Analyze the ITER_GAP and ITER_LEN information provided by
 8  all applications recorded in the given alog file.
 9
10  Standard Arguments:
11  file.alog - The input logfile.
12
13  Options:
14  -h,--help      Displays this help message
15  -v,--version  Displays the current release version
16
17  Further Notes:
18  See also: alogscan, alogrm, alogclip, alogview, aloggrep

```

8.2 Example Output from the alogiter Tool

The output shown in Listing 3 was generated from the `alpha.alog` file generated by the Alpha example mission, at time warp 20.

Listing 8.3: Example `alogiter` output applied to the `alpha.alog` file.


```

1 $ alogiter alpha.alog
2
3 Processing on file : MOOSLog_22_4_2015_13_25_19.alog
4
5   AppName      GAP      GAP      PCT      PCT      PCT
6   -----      - - - -  - - - -  - - - -  - - - -  - - - -
7   PHELMIVP     1.26     1.11     0.005    0.000    0.000
8   PMARINEVIEWER 1.10     1.07     0.000    0.000    0.000
9   PNODEREPORTER 1.25     1.15     0.008    0.000    0.000
10  UPROCESSWATCH 1.26     1.12     0.014    0.000    0.000
11  USIMMARINE    1.27     1.12     0.009    0.000    0.000
12
13
14  AppName      LEN      LEN      PCT      PCT      PCT      PCT
15  -----      - - - -  - - - -  - - - -  - - - -  - - - -
16  PHELMIVP     0.08     0.04     0.000    0.000    0.000    0.000
17  PMARINEVIEWER 0.00     0.00     0.000    0.000    0.000    0.000
18  PNODEREPORTER 0.01     0.00     0.000    0.000    0.000    0.000
19  UPROCESSWATCH 0.01     0.00     0.000    0.000    0.000    0.000
20  USIMMARINE    0.02     0.00     0.000    0.000    0.000    0.000
21
22
23 Mission Summary
24 -----
25 Collective APP_GAP: 1.11
26 Collective APP_LEN: 0.01

```

9 The alogsplit Tool

The `alogsplit` tool will split a given `.alog` file into a directory containing a file for each MOOS variable found in the `.alog` file. This is essentially the first stage of pre-processing done at the outset of launching the `alogview` tool. It is implemented here as a stand-alone app to be used for purposes other than `alogview`. It may also be useful as a command-line tool for preparing multiple `.alog` files from a shell script well before the first time they are used in `alogview`.

This is a new tool in Release 15.4 coinciding with the major re-write of the `alogview` tool also released in 15.4.

9.1 Naming and Cleaning the Auto-Generated Split Directories

The name of the *split directory* created by `alogsplit` is determined automatically from the `.alog` filename. For a file name `alpha.alog`, the directory created will be `alpha_alvtmp/` by default. This can be overridden with the command line switch `--dir=my_dirname`. The fairly distinctive `_alvtmp` suffix was chosen to facilitate cleaning these auto-generated temporary directories with a simple shell script, `alv_rm`:

```
#!/bin/bash
find . -name '*_alvtmp' -print -exec rm -rfv {} \;
```

The above script is found in the `moos-ivp/bin` directory and will remove (without prompting for confirmation) all split directories in the current directory and sub-directories.

9.2 Command Line Usage for the alogsplit Tool

```
$ alogsplit --help or -h
```

Listing 9.4: Command line usage for the `alogsplit` tool.

```
1 $ alogsplit -h
2
3 Usage:
4   alogsplit in.alog [OPTIONS]
5
6 Synopsis:
7   Split the given alog file into a directory, within which
8   each MOOS variable is split into it's own (klog) file
9   containing only that variable. The split will also create
10  a summary.klog file with summary information.
11
12  Given file.alog, file_alvtmp/ directory will be created.
13  Will not overwrite directory if previously created.
14  This is essentially the operation done at the outset of
15  launching the alogview applicaton.
16
17 Standard Arguments:
18   in.alog - The input logfile.
19
20 Options:
21   -h,--help      Displays this help message
22   -v,--version   Displays the current release version
```

```

23 --verbose      Show output for successful operation
24 --dir=DIR     Override the default dir with given dir.

```

9.3 Example Output from the alogsplit Tool

The output shown in Listing 5 was generated from the `alpha.alog` file generated by the Alpha example mission.

Listing 9.5: Example `alogsplit` directory applied to the `alpha.alog` file.

```

1 $ alogsplit alpha.alog
2
3 APPCAST.klog                IVPHELM_CPU.klog           NODE_REPORT_LOCAL.klog
4 APPCAST_REQ.klog           IVPHELM_CREATE_CPU.klog   PHELMIVP_ITER_GAP.klog
5 APPCAST_REQ_ALL.klog      IVPHELM_DOMAIN.klog      PHELMIVP_ITER_LEN.klog
6 APPCAST_REQ_ALPHA.klog    IVPHELM_IPF_CNT.klog     PLOGGER_CMD.klog
7 BHV_IPF_waypt_return.klog  IVPHELM_ITER.klog        PMARINEVIEWER_ITER_GAP.klog
8 BHV_IPF_waypt_survey.klog  IVPHELM_LIFE_EVENT.klog   PMARINEVIEWER_ITER_LEN.klog
9 CYCLE_INDEX.klog          IVPHELM_LOOP_CPU.klog     PMV_CONNECT.klog
10 CYCLE_INDEX_SURVEYING.klog IVPHELM_MODESET.klog      PNODEREPORTER_ITER_GAP.klog
11 DB_CLIENTS.klog           IVPHELM_REGISTER.klog    PNODEREPORTER_ITER_LEN.klog
12 DB_EVENT.klog             IVPHELM_STATE.klog       PROC_WATCH_EVENT.klog
13 DB_QOS.klog               IVPHELM_STATEVARS.klog   PROC_WATCH_FULL_SUMMARY.klog
14 DB_RWSUMMARY.klog        IVPHELM_SUMMARY.klog     PROC_WATCH_SUMMARY.klog
15 DB_TIME.klog              LOGGER_DIRECTORY.klog    PROC_WATCH_TIME_WARP.klog
16 DB_UPTIME.klog           MOOS_DEBUG.klog          RETURN.klog
17 DEPLOY.klog              MOOS_MANUAL_OVERRIDE.klog SIMULATION_MODE.klog
18 DESIRED_HEADING.klog     NAV_DEPTH.klog           TRUE_X.klog
19 DESIRED_RUDDER.klog      NAV_HEADING.klog         TRUE_Y.klog
20 DESIRED_SPEED.klog       NAV_HEADING_OVER_GROUND.klog UPROCESSWATCH_ITER_GAP.klog
21 DESIRED_THRUST.klog      NAV_LAT.klog             UPROCESSWATCH_ITER_LEN.klog
22 HELM_MAP_CLEAR.klog      NAV_LONG.klog            USIMMARINE_ITER_GAP.klog
23 IVPHELM_ALLSTOP.klog     NAV_PITCH.klog           USIMMARINE_ITER_LEN.klog
24 IVPHELM_ALLSTOP_DEBUG.klog NAV_SPEED.klog           USM_DRIFT_SUMMARY.klog
25 IVPHELM_BHV_ACTIVE.klog  NAV_SPEED_OVER_GROUND.klog USM_FSUMMARY.klog
26 IVPHELM_BHV_CNT.klog     NAV_X.klog               VISUALS.klog
27 IVPHELM_BHV_CNT_EVER.klog NAV_Y.klog               summary.klog
28 IVPHELM_BHV_IDLE.klog    NAV_YAW.klog
29 IVPHELM_BHV_RUNNING.klog NAV_Z.klog

```

Notice the `summary.klog` file on line 27. It contains some meta information gathered during the split process that is useful for `alogview` in fetching information at run time.

10 The `alogpare` Tool

The `alogpare` tool is a utility for pruning alog files by removing certain alog entries outside certain time windows. The time windows are defined by a user-defined time duration around the entries of further user-defined variables in the log file. The idea is that some robot missions have events of interest, e.g., a near collision event, where retaining all data just before and after the event is critical to analyzing what may have gone wrong. Perhaps certain high data rate log entries outside these critical event windows may be removed without any loss in utility to the users. In some cases this reduction in logged data may dramatically ease the archiving of these log files.

This was a new tool in Release 17.7 but was not documented until the following release.

10.1 Mark Variables Define Events of Interest

A *mark variable* is a MOOS variable provided to `alogpare` on the command line to indicate an event of interest. From the perspective of `alogpare`, the value of the mark variable does not matter. One or variables may be provided. For example:

```
$ alogpare --markvars=ENCOUNTER,NEAR_MISS
```

The `alogpare` utility will make an initial pass through the alog file and make note of each instance of a mark variable. A window of time, given by the command line parameter `--pare_window`, will be associated around each instance of a mark variable. If windows overlap, that's fine. The duration of the pare window is 30 seconds by default, even split in time before and after the mark event. This may be adjusted on the command line. For example:

```
$ alogpare --markvars=ENCOUNTER --pare_window=60
```

The `alogpare` utility will make a second pass through the alog file pruning log entries *outside* the pare windows, based on variables on the pare list.

10.2 The Pare List of Variables to be Pared

Variables on the *pare list* indicate which lines of an alog file are to be removed, outside of pare windows. The pare list is defined on the command line with:

```
$ alogpare --markvars=ENCOUNTER --pare_window=60 --parevars=BHV_IPF,BIG_ENTRY
```

Typically these variables constitute relatively large portions of an alog file, and provide little value outside the pare windows.

10.3 The Hit List of Variables to be Removed Completely

The `alogpare` utility also provides the means for removing named variables outright, regardless of where they occur relative to a pare window. These variables are on the *hit list*. For example:

```
$ alogpare --markvars=ENCOUNTER --parevars=BHV_IPF --hitvars=ITER_GAP
```

This functionality is also achieved with the `alogrm` utility, and is provided in this tool just as a convenience.

10.4 Command Line Usage for the `alogpare` Tool

```
$ alogpare --help or -h
```

Listing 10.6: Command line usage for the `alogpare` tool.

```
1 $ alogpare -h
2
3 Usage:
4 alogpare .alog [out.alog] [OPTIONS]
5
6 Synopsis:
7 Pare back the given alog file in a two-pass manner.
8 First pass detects events defined by given mark vars.
9 The second pass removes lines with vars on the pare
10 list if they are not within pare_window seconds of
11 an event line. It also removes lines with vars on the
13 hitlist unconditionally. Latter could also be done
15 with alogrm.
16 The original alog file is not altered.
17
18 Options:
19 -h,--help           Displays this help message
20 -v,--version        Display current release version
21 --verbose           Enable verbose output
22 --markvars=<L>      Comma-separated list of mark vars
23 --hitvars=<L>       Comma-separated list of hit vars
24 --parevars=<L>      Comma-separated list of pare vars
25 --pare_window=<N>  Set window to N seconds (default 30)
26
27 Examples:
28 alogpare --markvars=ENCOUNTER --parevars=BHV_IPF
29           original.alog smaller.alog
30 alogpare --markvars=ENCOUNTER
31           --parevars=BHV_IPF,VIEW_*
32           --hitvars=*ITER_GAP,*ITER_LEN,DB_QOS
33           --pare_window=10
34           original.alog smaller.alog
35
36 Further Notes:
37 (1) The order of alogfile args IS significant.
38 (2) The order of non alogfile args is not significant.
```

10.5 Planned additions to the `alogpare` Utility

- Soft parevars: removing perhaps every other entry outside pare window. Or remove success entries with identical values outside the pare window.
- Separate specification for pare_window time. Currently the window is split evenly around the mark event. Some user may want more control.
- Pattern matching: Add support for specifying sets of variables with simple wildcard prefix or suffix, e.g., `NAV_*` or `*_REPORT`.

11 The alogcd Tool

The `alogcd` tool is a utility for scanning a given alog file and tallying the number of encounters, near misses, and collisions. This utility works under the assumption that another utility had been running during the mission, and monitoring encounters, near misses and collisions. It assumes that these three events were separately noted with MOOS variables that also indicate the closest point of approach (CPA) range for each event. And it also assumes that these three variables were logged in the alog file.

The `alogcd` utility uses the MOOS variables `ENCOUNTER`, `NEAR_MISS`, and `COLLISION`. For now, these three variables are hard-coded in this utility. The `uFldCollisionDetect` utility is one utility capable of generating this kind of output. If there is collision to report, the report will also show the CPA value for the worst collision encounter.

An example run may produce output similar to:

```
$ alogcd file.alog

7,686 total alog file lines.

=====
Collision Report:
=====
Encounters: 27 (avg 16.93 m)
Near Misses: 6 (avg 10.18 m)
Collisions: 3 (avg 5.55 m)
Collision Worst: 3.87
```

11.1 Producing a Time-Stamped file of Collisions and Near Misses

The near misses and collisions are the real events of interests, and if the standard summary report is not enough, a time stamped list of each near miss and collision may be written to a file, if the `--tfile=filename` parameter is provided. For example, the six near misses and three collisions reported above could be written to file with:

```
$ alogcd file.alog --tfile=myfile
$ cat myfile
69.149,COLLISION,5.17
231.826,NEAR_MISS,9.41
351.374,NEAR_MISS,10.33
556.815,NEAR_MISS,10.35
592.976,NEAR_MISS,9.69
792.884,COLLISION,3.87
1065.484,NEAR_MISS,11.51
1129.862,COLLISION,7.61
1275.018,NEAR_MISS,9.82
```

The first column is the timestamp from the alog file, the second column is the type of encounter (near miss or collision), and the third column is the CPA distance for that encounter.

11.2 The Terse Output Option

For a super terse, one line report, use the following, which produces the below output for the same alog file as in the example above:

```
$ alogcd file.alog
27/6/3
```

27 encounters, 6 near misses, 3 collisions.

11.3 Command Line Return Values

The ultimate terse output is none at all! In this case we're only interested in the return value of `alogcd`. This can be used for example in a shell script to launch a series of simulations, altering the configuration parameters until no collisions are detected. The following (integer) return values are implemented:

- [0]: The alog file was found and readable, encounters were indeed reported, and no collisions were reported. The success condition.
- [1]: The alog file was not found or it was not readable.
- [2]: The alog file was indeed found and was readable, but sadly, collisions were reported.
- [3]: The alog file was indeed found and was readable, and no collisions were reported, but no encounters were reported either. Something is amiss. Either the vehicles never even got close enough to each other to constitute an encounter, or a monitoring app like `uFldCollisionDetect` was not even running.

11.4 Command Line Usage for the alogcd Tool

```
$ alogcd --help or -h
```

Listing 11.7: Command line usage for the `alogcd` tool.

```
1 $ alogcd -h
2
3 Usage:
4   alogcd .alog [OPTIONS]
5
6 Synopsis:
7   Scan an alog file for collision detection reports.
8   Tally the totals and averages, and optionally create
9   a file holding all the timestamps of events.
10
11  By default, it scans for events defined by postings
12  to the following three MOOS variables:
13
14  (1) COLLISION
15  (2) NEAR_MISS
16  (3) ENCOUNTER
17
18 Options:
19  -h,--help      Displays this help message
20  -v,--version   Display current release version
```

```
21  -t,--terse      Write terse output.
22
23  Returns:
24  0 if alog file ok, encounters detected, no collisions.
25  1 if alog file not ok, unable to open.
26  2 if alog ok, but collisions were detected
27  3 if alog ok, no collisions or encounters detected
```

11.5 Planned additions to the alogcd Utility

- Allow the key MOOS variables to be provided as parameters, rather than fixed to `ENCOUNTER`, `NEAR_MISS`, and `COLLISION`.
- Support cmd line option like `--collision_count` which produces the integer value of collision counts as the command line return value. Perhaps the same for `--near_miss_count` or `encounter_count`.

12 The alogcat Tool

The `alogcat` tool is a utility for concatenating a given set of alog files into a new single alog file. Recall that each alog file has a header at the beginning of the file with meta information. This includes the starting timestamp which allows all further timestamps to be relative to the starting time. So to concatenate alog files, we cannot simply just append one file onto the end another. If that were done, there would be multiple header blocks in the file and different blocks of data with timestamps relative to different start times.

Why would we need this tool? In certain field exercises, occasionally an operator may decide to stop the mission (killing the MOOS community), and restart with perhaps a slight modification in an important parameter. In such cases, two sets of log files will be produced, one from before the restart and one from after. The two of them may constitute a valid mission log file, but they are now split into two. The `alogcat` utility can be used for merging them back into one.

The `alogcat` tool performs a proper concatenation. First, it determines the chronological ordering of the provided alog files. It will use the header block and starting time of the earliest file. For the remaining files, (a) the relative time to the first file is calculated, (b) the header block of the older files is removed, (c) the log entries of the older file are appended to the end of the newer file with timestamps appropriately adjusted along the way.

An example run may produce output similar to:

```
$ alogcd file.alog

7,686 total alog file lines.

=====
Collision Report:
=====
Encounters: 27 (avg 16.93 m)
Near Misses: 6 (avg 10.18 m)
Collisions: 3 (avg 5.55 m)
Collision Worst: 3.87
```

12.1 Producing a Time-Stamped file of Collisions and Near Misses

The near misses and collisions are the real events of interests, and if the standard summary report is not enough, a time stamped list of each near miss and collision may be written to a file, if the `--tfile=filename` parameter is provided. For example, the six near misses and three collisions reported above could be written to file with:

```
$ alogcd file.alog --tfile=myfile
$ cat myfile
69.149,COLLISION,5.17
231.826,NEAR_MISS,9.41
351.374,NEAR_MISS,10.33
556.815,NEAR_MISS,10.35
592.976,NEAR_MISS,9.69
792.884,COLLISION,3.87
1065.484,NEAR_MISS,11.51
1129.862,COLLISION,7.61
1275.018,NEAR_MISS,9.82
```

The first column is the timestamp from the alog file, the second column is the type of encounter (near miss or collision), and the third column is the CPA distance for that encounter.

12.2 The Terse Output Option

For a super terse, one line report, use the following, which produces the below output for the same alog file as in the example above:

```
$ alogcd file.alog
27/6/3
```

27 encounters, 6 near misses, 3 collisions.

12.3 Command Line Return Values

The ultimate terse output is none at all! In this case we're only interested in the return value of `alogcd`. This can be used for example in a shell script to launch a series of simulations, altering the configuration parameters until no collisions are detected. The following (integer) return values are implemented:

- [0]: The alog file was found and readable, encounters were indeed reported, and no collisions were reported. The success condition.
- [1]: The alog file was not found or it was not readable.
- [2]: The alog file was indeed found and was readable, but sadly, collisions were reported.
- [3]: The alog file was indeed found and was readable, and no collisions were reported, but no encounters were reported either. Something is amiss. Either the vehicles never even got close enough to each other to constitute an encounter, or a monitoring app like `uFldCollisionDetect` was not even running.

12.4 Command Line Usage for the alogcd Tool

```
$ alogcd --help or -h
```

Listing 12.8: Command line usage for the `alogcd` tool.

```
1 $ alogcd -h
```

```

2
3 Usage:
4   alogcd .alog [OPTIONS]
5
6 Synopsis:
7   Scan an alog file for collision detection reports.
8   Tally the totals and averages, and optionally create
9   a file holding all the timestamps of events.
10
11  By default, it scans for events defined by postings
12  to the following three MOOS variables:
13
14  (1) COLLISION
15  (2) NEAR_MISS
16  (3) ENCOUNTER
17
18 Options:
19   -h,--help      Displays this help message
20   -v,--version   Display current release version
21   -t,--terse    Write terse output.
22
23 Returns:
24   0 if alog file ok, encounters detected, no collisions.
25   1 if alog file not ok, unable to open.
26   2 if alog ok, but collisions were detected
27   3 if alog ok, no collisions or encounters detected

```

12.5 Planned additions to the alogcd Utility

- Allow the key MOOS variables to be provided as parameters, rather than fixed to `ENCOUNTER`, `NEAR_MISS`, and `COLLISION`.
- Support cmd line option like `--collision_count` which produces the integer value of collision counts as the command line return value. Perhaps the same for `--near_miss_count` or `encounter_count`.