Help Topic: Launching a Mission with pAntler

Spring 2020

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

Launching a Mission with pAntler

In theory a set of N MOOS applications may be launched from N terminal windows, but this is cumbersome in practice. The `pAntler` tool allows this to be done from a single mission file. In this file, a block of lines declares all the apps to be launched with one invocation of `pAntler`.

Where to get more information:

- `pAntler`: `http://oceanai.mit.edu/ivpman/apps/pAntler`

Basic pAntler Usage

The Antler block is typically the first configuration block in a .moos file, declared with `ProcessConfig = ANTLER` as below. The `MSBetweenLaunches` parameter specifies the number of milliseconds between launching processes. Each line thereafter specifies an app to be launched and whether a dedicated console window should be opened for the application.

```
ProcessConfig = ANTLER
{
  MSBetweenLaunches = 200

  Run = MOOSDB       @ NewConsole = true/false
  Run = AnotherApp   @ NewConsole = true/false
  ...
  Run = AnotherApp   @ NewConsole = true/false
}
```

Further options exist beyond the vanilla launch configuration described above, including (a) the ability to launch a given app under an aliased name, (b) specifying command-line arguments to an app at launch time and more. See the documentation.

An Example: Launching the MOOSDB along with uXMS

In the example below we use `pAntler` to launch the `MOOSDB` and the `uXMS` scope from a single mission file. The user preferences for `uXMS` are provided in its configuration block. Type `uXMS --example` on the command line for further options if you're curious.

Note: in this example, the launch process should result in a new `xterm` window opening. If you are running on GNU/Linux, the `xterm` should be available by default. If you are running MacOS, you may need to install xterm. If you type xterm on the MacOS Terminal command line and you receive a "command not found" error, then you may need to install xterm.

https://www.xquartz.org

You may need to log out and log back in for this to take effect.

Your goals in this part are:

1. Create a copy of the example mission file shown in Listing 1 below and save it locally as db_and_uxms.moos. (hint: the easiest way to do this is to just invoke the wget expression on the top line of this file. This will pull the file down from the server into your current directory.) The mission may be launched from the command-line with:

   ```
   $ pAntler db_and_uxms.moos
   ```

   This should open a new console window for uXMS displaying the variables posted by the DB, with the (S)ource and (T)ime columns expanded, but not the (C)ommunity column.

2. Modify the uXMS configuration block in the .moos file to configure uXMS to keep a history of the DB_UPTIME variable. To see configuration options for uXMS, type:

   ```
   $ uXMS --example
   ```

   Once you have launched uXMS with the new configuration, type 'z' to toggle in and out of history mode.

3. Modify the db_and_uxms.moos file to launch a new terminal window for the MOOSDB in addition to the uXMS application.

Listing 0.1: A simple mission file.

```
// (wget http://oceanai.mit.edu/2.680/examples/db_and_uxms.moos)
ServerHost = localhost
ServerPort = 9000
Community  = alpha

ProcessConfig = ANTLER
{
  MSBetweenLaunches = 200

  Run = MOOSDB     @ NewConsole = false
  Run = uXMS       @ NewConsole = true
}

ProcessConfig = uXMS
{
  AppTick   = 4
  CommsTick = 4

  VAR            = DB_CLIENTS, DB_UPTIME, DB_TIME
  DISPLAY_SOURCE = true
  DISPLAY_TIME   = true
  COLOR_MAP      = DB_CLIENTS, red
}
```