Launching MOOS Processes and Mission scripts with `pAntler`

Paul Newman

September 28, 2007

Abstract

This document tell you how to use the application `pAntler` to launch multiple MOOS processes. This is usefultool for starting up a whole bunck of processes all of which sahre a single configuration file.

1 Introduction

The process `pAntler` is used to launch/create a MOOS community. It is very simple and very useful. It reads from its configuration block a list of process names that will constitute the MOOS community. Each process to be launched is specfied with a line with the syntax

`Run` = *procname* [@ `NewConsole` = *true/false*] [∼ `MOOSName`]

The optional console parameter specifies whether the named process should be launched in a new window (an xterm in Unix or cmd-prompt in NT derived platforms). Each process launched is passed the mission file name as a command line argument. When the processes have been launched `pAntler` waits for all of the community to exit and then quits itself.

1.1 Running Multiple Instances of a Process

The optional `MOOSName` parameter allows MOOSProcesses to connect to the `MOOSDB` under a specified name. For example a vehicle may have two GPS instruments onboard. Now by default `iGPS` may register it existence with the `MOOSDB` under the name "iGPS". By using this syntax multiple instances of the executable `iGPS` can be run but each connects to a the `MOOSDB` using a different name.

```
Run = iGPS @ NewConsole = true ∼ iGPSA
Run = iGPS @ NewConsole = true ∼ iGPSB
```

would launch two instances of `iGPS` registering under "iGPSA" and "iGPSB" respectively. Note there would need to be *two* GPS configuration blocks in the mission file – one for each and the processnames would be "iGPSA" and "iGPSB"

2 Launching Missions

`pAntler` provides a simple and compact way to start a MOOS session. For example if the desired mission file is *Mission.moos* then executing

```
pAntler Mission.moos
```

will launch the required processes for the mission. Of course a sensibly designed mission will not actually start to do anything until a human (via `iRemote`) has confirmed a good working status of the processes involved (eg `pNav`) and actively hands control over to the Helm.

2.1 I/O Redirection - Deployment

As already mentioned, frequently `iRemote` , displayed on a remote machine, will be the only interface a mission pilot has to the MOOS community. We must ask the question - "where does all the IO from other processes go to prevent I/O blocking?". One answer to this is I/O redirection and backgrounding MOOS processes - a simple task in unix derived systems [1]/

Running `pAntler` in the following fashion followed by a manual start up of `iRemote` is the recommended way of running MOOS in the field using a serial port login.

1. `pAntler` *mission.moos* > ptyZ0 > /dev/null &
2. `iRemote` *mission.moos*

This redirection of `iRemote` is encapsulated in the `moosbg` script included with the MOOS installations. In the case of an AUV the interface can only be reached through in-air wireless communications, which will clearly disappear when the vehicle submerges but will gracefully re-connect when surfacing(not so easy to do with a PPP or similar link).

[1] some OS are good for development others for running...