

A SIMPLE SYSTEM FOR LOGGING MESSAGES TO A WINDOW

ERWIN KALVELAGEN

ABSTRACT. This document describes a simple way to write messages to a window from a GAMS job. Messages written to the standard log file or to the listing file can be difficult to keep track of. This system will allow to create one or more windows that can receive messages from a running GAMS job.

1. INTRODUCTION

For more complex, long running GAMS jobs it is often difficult to monitor progress by looking at the log file. Too many messages can obscure information about how far the job has proceeded. Some ideas have been explored by users facing this problem in the past, such as updating the title string of a command window. In this document we describe a small system that can create one or more windows, where we can send messages to from a running GAMS job.

2. ARCHITECTURE

The system consists of two parts: a *server* which displays messages to a window and a *client* which can send messages. The communication is implemented through *named pipes*, a facility available under Windows NT and XP. A named pipe is pseudo filename of the form `\\.\pipe\name`. The server will read from this pipe and the clients write to this file. In fact, a client could be as simple as a GAMS put statement:

```
file pipe '\\.\pipe\test';  
putclose pipe "Hello world";
```

Unfortunately, two issues make this approach not feasible:

- A GAMS bug will not allow you to write to `\\.\pipe\test`. The file name is interpreted as `\\pipe\name` which is not the same.
- Synchronization issues may prevent the connection to the pipe. A well-behaved client will invoke the API call `WaitNamedPipe` to wait for an available instance of a pipe.

Instead we developed a dedicated client program `wlog.exe` which can send message provided on the command line:

```
execute '+wlog pipename=test "Hello world";
```

or from put files:

```
file mess '/test.txt';  
putclose mess "Hello world";  
execute '+wlog pipename=test @test.txt';
```

Date: May, 2004.

The server is automatically launched by `wlog.exe` if it is not running already.

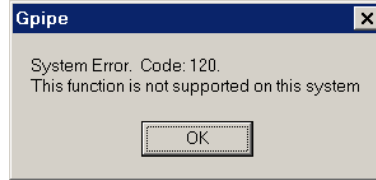


FIGURE 1. Error message when running the server on Windows 98/ME

The *named pipe* API calls will only work on windows NT and XP systems. When the server is started on a windows 98/ME system, an error message will be issued as shown in figure 1.

3. EXAMPLES

time	description
6/10/2004 6:19:48 PM	starting
6/10/2004 6:19:48 PM	iteration i1
6/10/2004 6:19:48 PM	iteration i2
6/10/2004 6:19:48 PM	iteration i3
6/10/2004 6:19:48 PM	iteration i4
6/10/2004 6:19:48 PM	iteration i5
6/10/2004 6:19:48 PM	iteration i6
6/10/2004 6:19:48 PM	iteration i7
6/10/2004 6:19:48 PM	iteration i8
6/10/2004 6:19:48 PM	iteration i9
6/10/2004 6:19:48 PM	iteration i10
6/10/2004 6:19:48 PM	done

FIGURE 2. Log window for example model

The model:

```
execute 'wlog pipename=test "starting";
file f /data.txt/;

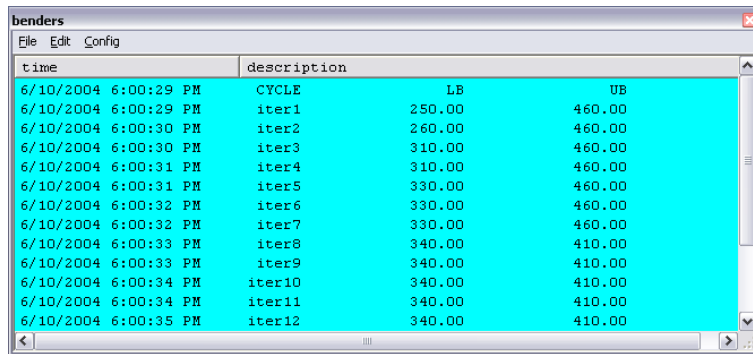
set i /i1*i10/;
loop(i,
  putclose f "iteration ",i.tl;
  execute 'wlog pipename=test @data.txt';
);

execute 'wlog pipename=test "done";
```

will generate a log window like shown in figure 2.

If you want to write tabular information it is useful to specify a non-proportional font such as *Courier New*. An example is the progress of the Benders Algorithm[1], where we want to monitor lower and upper-bounds, as shown in figure 3.

The statements involving writing to the log window are as follows:



The screenshot shows a window titled 'benders' with a menu bar (File, Edit, Config) and a table of log data. The table has two columns: 'time' and 'description'. The 'description' column is further divided into 'CYCLE', 'LB', and 'UB' sub-columns. The log shows 12 iterations of data, with the LB and UB values converging over time.

time	description
6/10/2004 6:00:29 PM	CYCLE LB UB
6/10/2004 6:00:29 PM	iter1 250.00 460.00
6/10/2004 6:00:30 PM	iter2 260.00 460.00
6/10/2004 6:00:30 PM	iter3 310.00 460.00
6/10/2004 6:00:31 PM	iter4 310.00 460.00
6/10/2004 6:00:31 PM	iter5 330.00 460.00
6/10/2004 6:00:32 PM	iter6 330.00 460.00
6/10/2004 6:00:32 PM	iter7 330.00 460.00
6/10/2004 6:00:33 PM	iter8 340.00 410.00
6/10/2004 6:00:33 PM	iter9 340.00 410.00
6/10/2004 6:00:34 PM	iter10 340.00 410.00
6/10/2004 6:00:34 PM	iter11 340.00 410.00
6/10/2004 6:00:35 PM	iter12 340.00 410.00

FIGURE 3. Benders' lower and upper-bounds

```

file wlog /wlog.txt/;
wlog.tj = 1;
wlog.lj = 1;
put wlog;
putclose "CYCLE":6,"LB":18,"UB":18/;
execute 'wlog pipe=Benders @wlog.txt';

```

and

```

putclose wlog, iter.tl:6,LB:18,UB:18;
execute 'wlog pipe=Benders @wlog.txt';

```

4. OPTIONS

The main menu has an *Edit* menu. The two entries are:

- Clear. Clear the list: existing lines will be erased.
- Autoscroll. Whether the window should scroll to the bottom when a new message entry is added. If you want to inspect a part of the window you may want to disable this.

The *Config* menu has the following settings

- Color. Background color of the log window.
- Font. Font used in the log window.
- Stay on top. Whether the form style of the log window is *stay-on-top*. If this option is checked the log window will always be on top of other windows.
- Timestamp. If this option is checked a column with the date and time is added to the log. This is the default.

The settings and the location and size of the window will be saved in an *.ini* file in the GAMS system directory, so they will be reloaded if the server program is started again with the same pipe name.

REFERENCES

1. Erwin Kalvelagen, *Benders Decomposition with GAMS*, <http://www.gams.com/~erwin/benders/benders.pdf>, December 2002.

GAMS DEVELOPMENT CORP., WASHINGTON DC
E-mail address: erwin@gams.com