DebuggingCode

Contents

Using core files

When code crashes on BGL, usually you'll get `core.<process id>` files, which are plain-text stack traces. You can figure out where your code crashed by using `addr2line -e <exe> <address>`. You get the addresses from the "function call chain:" section at the end of the core file.

```
function call chain:
0x001521f4
0x00152148
0x00170604
0x00100a24
0x00100158
0xfffffffc
```

Jim Edwards wrote a perl script to automatically spit out a full stack trace for a given binary; it is located at `/contrib/bgl/bin/decode.pl`. **You need to compile and link with the '-g' flag to get the proper routine names**.

Example:

```
decode.pl test.exe core.0
```

Monitoring stack and heap usage (memmon)

The memmon library is at `/contrib/bgl/lib/libmemmon.rts.a`, with a README at `/contrib/bgl/memmon/README`.

Memmon allows you to trace and view stack and heap usage in your code. These functions are provided:

C	Fortran
`void memmon_trace_on(int *rank_p)`	
`void memmon_trace_off(int *rank_p)`	
`void memmon_print_usage()`	

Compile your code with these flags, depending on the compiler you are using:

- xlc/xlf: `-qdebug=function_trace`
  gcc: `-finstrument-functions`

Link with `-L/contrib/bgl/lib -lmemmon.rts`

If you just link with memmon and do not add any of the memmon_ function calls, memmon will watch your memory usage and exit if the stack overwrites the heap.

If you add calls to `memmon_trace_on` and `memmon_trace_off` in your code, memmon will print memory usage at the entry and exit of each funtion surrounded by the `memmon_trace_on` and `off` functions.

```
entering somefunc, min free mem: 508.58MB, stack min: 0x1ffaa728 (somefunc;entry), heap max: 0x00316000
(somefunc;entry)
exiting somefunc, min free mem: 508.58MB, stack min: 0x1ffaa728 (somefunc;entry), heap max: 0x00316000
(somefunc;entry)
```

Calling `memon_print_usage()` anywhere in your code prints out a message like this:

```
Min Free Memory: 508.58MB, stack min: 0x1ffaa728 (somefunc;entry), heap max: 0x00316000 (somefunc;entry)
```

Note that memmon shows the maximum (not the current) stack and heap usage in the program at the point where the routine is called.