

Debuggers

Mary Kate Trost

Debugger

Advantages/Disadvantages

- Advantages
 - Single step through the code
 - Stop execution at a given point to investigate where it goes and what the values are
 - Attach to already running program
- Disadvantages
 - Not running real-time, so may not expose all problems

DDD

Deugging tools:

Run: start the program

Interreupt: stop the program from running

Step: Go into the function call (or go to next line of code)

Next: Go over a function call (execute it, but do not step into it)

Finish: Continue execution until the end of the current method

Cont: Continue execution until the next breakpoint or the end of the program is reached.

Kill: stop the program from running.

DDD: /home/mktrost/pipeline/bam/SamRecord.cpp

File Edit View Program Commands Status Source Data Help

(): .length() Lookup Find Break Watch Print Display Plot Show Rotate Set Undo

Run Interrupt Step StepI Next NextI Until Finish Cont Kill Up Down

1: myStatus

```
enumStatusString = 0x64e4a0
myType            = SamStatus::SUCCESS
myMessage         = {...}
```

```
822 {
823     myStatus = SamStatus::SUCCESS;
824     if(myCigar.Length() == 0)
825     {
826         // 0 Length, means that it is in the
buffer, but has not yet
827         // been synced to the string, so do
the sync.
828         parseCigarBinary();
```

```
--maxReportedErrors [100]
SortOrder : --so_flag, --so_coord, --so_query
```

Breakpoint 1, SamRecord::getCigar (this=0x7fffffffda90) at SamRecord.cpp:824
(gdb) print myCigar.Length()
\$1 = 4
(gdb) p/x myCigar.Length()
\$2 = 0x4
(gdb)

Δ \$2 = 0x4

Basic How To Use

- Bring up a file in the viewer:
 - | <filename>:<line#>
 - | SamRecord.cpp:1
 - | <class>::<method>
 - | SamRecord::getCigar
- Set a breakpoint
 - Use mouse right-click on the line number
 - Set Breakpoint (can set properties – break after hit X number of times, etc)
 - b <class>::<method>
- Attach to already running process
 - File->Attach to Process

Basic How To Use (cont)

- Run with options
 - On the command line (type run in place of your executable name):
 - run <options>
- Backtrace (see where you are in execution, look up/down the call stack):
 - Status->Backtrace
- See a variable's value:
 - Right click the variable in the source code window and click “Print” (or to keep it tracked, click “Display”)
 - On command line: p <variable>
 - In hex: p/x <variable>

Other Testing Advice

- Reduce test size from one that takes hours to one that is much quicker.
 - Reduce file sizes
 - Turn off unnecessary sections
- Write a set of automated tests that test the different cases so they can be re-run each time the library changes
- When you find a bug, write a test that exposes the bug (fails), fix the bug, rerun the test (succeeds)