Uso do GDB na infraestrutura ARM para MC404

Alexandre Medeiros

10 de maio de 2013

1 Introdução - O que é o GDB?

GDB, o depurador do projeto GNU, é um programa cujo objetivo é localizar bugs em outros programas, para isso ele possui diversas funcionalidades, como pausar a execução do programa em determinados pontos, examinar o valor de variáveis e até mesmo executar o programa instrução por instrução.

Figura 1: Logo do GDB

2 Como usar?

Antes de utilizar o GDB é necessário compilar o seu programa com as informações de depuração necessárias para que o GDB execute-o corretamente, disponibilizando informações sobre as variáveis, rótulos e instruções que estão sendo executadas.

Para isso, ao montar e ligar seu programa, basta adicionar a flag `-g`, logo, os comandos ficarão:

```
$ as arquivo.s -g -o arquivo.o
```

```
$ gcc arquivo1.o [arquivo2.o ...] -g -o programa
```

Agora, para utilizar o GDB para depurar seu programa, basta executar o comando:

```
$ gdb ./programa
```

3 Como depurar?

Agora que conseguimos abrir nosso programa no GDB, o que podemos fazer de bom? A princípio, seu programa pode ser executado normalmente no GDB, basta usar o comando:

```
(gdb) run [argumentos]
```

Com isso seu programa será executado com os argumentos passados para o comando `run`, note que esses argumentos são os mesmos utilizados para executar seu programa no terminal, como em

```
$ ./programa arquivo_entrada.txt > saida.txt
```

Usando apenas o `run` seu programa será executado até o fim, seja esse fim o fim de execução normal (`return 0;` na main) ou um erro como o temido Segmentation Fault.

No caso de um erro como o SegFault o GDB irá indicar em qual linha do seu código que aconteceu esse problema e lhe permitirá examinar os valores das variáveis nesse ponto.

3.1 Pausando a execução

Uma das principais funcionalidades do GDB é ter a capacidade de pausar a execução do seu programa em determinados pontos, esses pontos são conhecidos como breakpoints.

3.1.1 Breakpoints

Um breakpoint nada mais é do que um ponto no código que, antes de ser executado, o GDB pausa a execução e disponibiliza seu terminal para que você possa inspecionar a situação atual do seu programa.

Há diversas maneiras de se criar um breakpoint, você pode criá-lo a partir de uma linha do seu arquivo fonte, por exemplo, você deseja que o programa pause na instrução que está na linha 7 do arquivo main.s, você deve usar o comando

```
(gdb) break main.s:7
```

Também podemos criar um breakpoint numa função. Considere que queremos criar um breakpoint na função foo do arquivo bar.s, para isso, basta usar o comando

```
(gdb) break bar.s:foo
```

Observe que, caso haja apenas um arquivo fonte, o nome do arquivo é desnecessário em ambos comandos acima.

Quando não se precisa mais de um breakpoint, é conveniente removê-lo. Para remover um breakpoint, utiliza-se este comando:

```
(gdb) del NUMERO
```

Onde NUMERO representa o número do breakpoint que se deseja remover. Note que cada breakpoint possui um número que o identifica, esse número aparece ao criá-lo e sempre que o GDB pausa a execução nele.

3.2 Examinando dados

Agora que o programa está pausado e como que podemos examinar seu status?

Existem diversas maneiras de se verificar a situação atual do programa, pode-se investigar os valores nos registradores ou os valores de alguma variável específica.

3.2.1 Registradores

Quando se está depurando um programa em linguagem de montagem, uma informação muito importante é qual o valor nos registradores. Por isso, ela é consideravelmente simples de se conseguir, quando o GDB pausar num breakpoint, para descobrir quais os valores que estão nos registradores, basta utilizar o comando

```
(gdb) info registers
```

Esse comando listará os registradores pelo nome seguidos de duas colunas, sendo a primeira o valor que está no registrador no formato hexadecimal e na segunda o mesmo valor em decimal.

O comando `info`, além de obter as informações dos registradores, também pode ser usado para obter informações dos breakpoints que você criou, para isso, é só utilizar `info break`.

3.2.2 Variável

Também é possível verificar o valor de uma variável, para tal, deve-se utilizar o comando `print` que recebe como argumento o nome de uma variável, note que esse nome de variável pode ser o nome do rótulo associado à essa variável.

Observe também que no argumento do `print` pode-se usar diversos dos operadores de expressões usados em C, com por exemplo, soma, multiplicação, o operador unário `&`, que retorna o endereço de uma variável, até mesmo o operador unário `*` que considera o valor ao qual é aplicado como sendo o endereço de uma variável. Também há a possibilidade de se utilizar typecasts para os tipos comuns como `int` e `char`.

3.3 Continuando a execução

Depois de pausar a execução e verificar o valor de suas variáveis... e agora? Como voltar a executar o programa?

Há várias maneiras de se retomar a execução do programa, que podem ser passo a passo (instrução por instrução) ou executando o programa até terminar ou encontrar outro breakpoint.

Para executar o programa passo a passo, existem dois comandos: o `stepi` e `nexti`, a principal diferença entre os dois é que o `stepi` (step instruction) realmente executa tudo passo a passo até mesmo chamadas de funções, já o `nexti` (next instruction) executa instrução por instrução mas ao encontrar uma chamada de função executa a função como se fosse uma única instrução.

Além de executar instruções por instruções, também é possivel voltar a executar o programa normalmente, para isso existe o comando `continue` que retoma a execução do programa normalmente ate que ele acabe ou encontre algum breakpoint.

3.3.1 Mas onde eu estou agora?

É muito fácil durante a depuração do programa perder noção de em qual trecho do código você está, o GDB ao pausar a execução lhe mostra linha que ele irá executar, mas as vezes isso não é o suficiente, para isso existe um comando chamado `list`, esse comando lista um trecho do código que está sendo depurado. Se utilizado sem nenhum argumento, ele lista o trecho atual do programa, porém é possível fazê-lo listar alguma função específica ou até mesmo uma linha específica de algum arquivo, para isso, você deve utilizá-lo assim

```
(gdb) list [arquivo:]arg
```

onde o `arg` pode ser o número da linha ou o nome de uma função, note que o argumento opcional (`arquivo`) é o nome do arquivo do qual você deseja listar um trecho.

4 Abreviações de comandos

Como existem diversos comandos que são repetidos diversas vezes, o GDB possui alguns "atalhos" para os mesmo, esse atalho nada mais é do que o comando abreviado, veja na tabela 1 a lista das principais abreviações do GDB.

Comando	Abreviação
`run`	`r`
`break`	`b`
`info`	`i`
`registers`	`reg`
`print`	`p`
`stepi`	`si`
`nexti`	`ni`
`continue`	`c`
`list`	`l`

Tabela 1: Abreviação de alguns comandos do GDB

Além disso, no terminal do GDB, ao precionar a tecla Enter sem digitar nada, o GDB executa o último comando, então por exemplo, você está executando uma rotina passo a passo (com o `stepi`) e ao invés de digitar `stepi` a cada instrução, basta digitar o comando uma vez e apertar a tecla Enter sem digitar nada.

5 Mais informações

O GDB é um programa bem complexo e é difícil decorar todos seus comandos e isso não é esperado de ninguém, tanto que o próprio GDB possui comandos cuja função são ajudar as pessoas a lembrarem de outros comandos.

Os comandos de ajuda são:

- `help`
- `apropos`

Com esses dois comandos é possível relembrar a sintaxe de determinados comandos e comandos relacionados com certas funcionalidades. O `apropos` é utilizado para buscar na descrição dos comandos existentes alguma palavra chave, por exemplo, o comando:

```
(gdb) apropos breakpoint
```

Esse comando listará todos os comandos relacionados com breakpoints e uma descrição dos mesmos, nessa lista temos o comando `break`, que para ver como ele deve ser utilizado, basta usar o comando

```
(gdb) help break
```

que listará a sintaxe do comando junto de uma explicação do significado de cada argumento.

Além desses comandos, é muito fácil encontrar informações sobre o GDB pela internet, um bom lugar para começar é na Documentação oficial (`http://www.gnu.org/software/gdb`) em Inglês.