Integração do Mecanismo de Exceções ao EPOS

Felipe Zimmermann Homma
Guilherme Bertuol
Mateus Krepsky Ludwich
Tiê Teixeira Lima Penatti

9 de março de 2007

1 Objetivo

Integrar os mecanismos de manejo de exceções do `GCC` ao `EPOS`. Um segundo passo seria integrar esse mecanismo ao sistema de interrupções do `SO` de forma a possibilitar um tratamento unificado de interrupções e exceções.

Devido a total desorganização do grupo a segundo parte tornou-se inviável por falta de tempo.

Sendo assim o atual objetivo fica por fazer o manejo de exceções ser utilizado com o `EPOS`.

2 Trabalhos Iniciais

Procurando pela documentação e rastreando pelo código do `GCC` foi possível definir algumas estratégias de abordagem ao problema.

Existe uma documentação descrevendo todo o processo de tratamento de erros utilizado no `GCC`. O fato mais desanimador é de que esse sistema todo esta em estágio de desenvolvimento pesado, sem garantias de que o padrão permaneça o mesmo nas versões futuras.

2.1 ABI Itanium©

O sistema de tratamento de erros do `GCC` baseia-se na proposta da Intel©para o Itanium©. Ela divide o tratamento em três níveis, esta divisão torna possível portabilidade e interoperabilidade entre plataformas/linguagens/compiladores.

No primeiro nível é independente de linguagem de programação. Ele trata da busca busca dos tratadores através da pilha. É dividido em duas fases:

Busca: utiliza a "personality routine", que faz sucessivos *unwinds* da *stack* procurando por um tratador adequado para a exceção ocorrida.

Cleanup: novamente a "personality routine" (com *flags* diferentes da fase anterior) executa *unwinds* sucessivos até encontrar o stack frame onde está o tratador (*Landing Pad*) localizado na primeira fase.

O controle é então passado ao `Landing Pad`, que trata a exceção.

O segundo nível liga o `Unwind` à linguagem de programação (`C++` no caso). Nesse nível é definida a "personality routine" que pertence a biblioteca de tempo de execução do `C++`. E o terceiro nível especifica questões adicionais ao segundo, para que implementações distintas de `C++` possam compartilhar o mesmo sistema de tempo de execução que dá suporte ao tratamento de exceções.

2.2 Isolamento das rotinas necessárias

No `GCC` atual, as rotinas do segundo nível estão centralizadas (e relativamente isoladas) na biblioteca `libsupc++`, componente da biblioteca padrão de `C++`.

São oferecidas rotinas para a redefiniçao dos `handlers` de exceção. Isso permite desviar a execução para um código personalizado. Ainda são necessários mais estudos para verificar se esse é realmente o código para tratamento de qualquer exceção ou se é o recurso *default*.

3 Estado do trabalho

Embora o isolamento da biblioteca pareça simples, logo nos deparamos com problemas de ligação. Forma mapeados os seguintes problemas:

1. Manipulação de stream
2. Gerência de memória
3. Coisas Linux

O primeiro já foi isolado. Tratava-se de uma rotina de um recurso que nem estava sendo utilizado (`verbose_terminate`), foi removido diretamente do fonte.

Os de gerência de memória estão espalhados na forma de malloc e free (talvez outras rotinas, mas ainda não foram localizadas) em alguns arquivos (`eh_alloc.cc`, `eh_globals.cc`, `new_op.cc`, `new_opnt.cc`, `del_op.cc`, `del_opnt.cc`). Falta verificar as interdependências, mas aparentemente são reescritas para os operadores padrão new e delete, de forma que lancem as exceções e a alocação de recursos para o sistema de manejo de exceções.

As coisas linux estão relacionados com a `Unwind` e apesar das funções serem conhecidas, parecem não ter funções relativas no `EPOS` (tão pouco utilidade, basicamente ele percorre as bibliotecas de vínculo dinâmico do programa).

O próximo passo seria uma reescrita das partes dependentes do manejo de memória e depois a extirpação das coisas linux, talvez simplesmente pela extração.

Como procedimento para isolar a parte de exceções do resto da libgcc e pode-la integrar ao `EPOS` copiamos vários arquivos do diretório libsupc++ dos fontes do gcc para os diretórios `include/eh` e `src/eh` do `EPOS`. Criamos um `makefile` e adaptamos o `makedefs` para realizar a compilação. Fomos então compilando e eliminando as dependências na medida do possível. Deparamos-nos com alguns erros de linkedição no caminho, mas por fim resolvemos todos. Entretanto, ao excecutar a aplicação elaborada para testar as exceções, obtemos um erro de "out of memory"no momento em que se passa pela cláusula throw e uma exceção é lançada.