1
Introdução

Existem muitas situações onde é necessário ou interessante que haja interação entre programas escritos em diferentes linguagens. Um caso típico é o emprego de bibliotecas externas, como *toolkits* gráficos, APIs de acesso a banco de dados, ou até mesmo chamadas ao sistema operacional. Outro cenário ainda envolve aplicações desenvolvidas usando mais de uma linguagem de programação a fim de otimizar partes onde o desempenho é crítico ou permitir extensibilidade através de *scripts* escritos pelo usuário.

Independentemente da finalidade, a comunicação entre programas escritos em linguagens diferentes traz consigo uma série de questões de projeto, não apenas no desenvolvimento das aplicações, mas das linguagens em si. Há várias formas de se obter esse tipo de interoperabilidade, desde tradução de código de uma linguagem para outra até o uso de uma máquina virtual comum. Idealmente, entretanto, uma linguagem deve prover uma interface de acesso externo (*foreign language interface*, FLI) que permita ao programador receber e enviar tanto chamadas como dados para outra linguagem (Finne 1998). Entre os fatores que devem ser levados em consideração no desenvolvimento de tal interface estão as diferenças entre os sistemas de tipos, problemas de gerência de memória (como coleta de lixo e acesso direto a ponteiros) e modelos de concorrência. Além de lidar com diferenças semânticas, o projeto de uma interface entre linguagens envolve questões pragmáticas como o equilíbrio entre o isolamento seguro dos ambientes de execução, o desempenho e a simplicidade da API resultante.

Pode-se observar nas implementações existentes de FLIs um número de abordagens para estes problemas. De fato, FLIs de diferentes linguagens (ou mesmo de diferentes revisões de uma mesma linguagem) tendem a ser bastante distintas entre si. Ainda assim é possível traçar paralelos entre as técnicas utilizadas, uma vez que os problemas fundamentais que elas atacam são os mesmos.

Em função da popularidade da linguagem C e do suporte oferecido a ela pelos sistemas operacionais mais utilizados, grande parte das implementações de interfaces de acesso externo são, na prática, APIs para C. Além

disso, um modelo de interação entre linguagens que tem se mostrado especialmente relevante na atualidade é o que se dá entre linguagens compiladas tipadas estaticamente, como C, e linguagens interpretadas tipadas dinamicamente, tipicamente chamadas de linguagens de script, como defendido por Ousterhout (Ousterhout 1998). Estas duas categorias de linguagens possuem objetivos fundamentalmente diferentes. Linguagens estaticamente tipadas são usualmente implementadas visando alto desempenho e possuem um enfoque de mais baixo nível. Em contraste, linguagens de script tendem a ser implementadas como interpretadores ou máquinas virtuais e fazem uso extensivo de construções de alto nível, tais como listas e hashes, como sendo tipos básicos. Estas características complementares têm tornado popular o modelo de programação baseado em duas linguagens, onde uma linguagem de mais baixo nível é usada para o desenvolvimento de componentes que são conectados através de uma linguagem de mais alto nível.

1.1 Objetivo

Este trabalho discute as principais questões envolvendo o projeto de APIs para integração de ambientes de execução de linguagens de script em aplicações C. Apresentamos os principais problemas enfrentados na interação entre código executando em um ambiente com características inerentemente dinâmicas como o de uma linguagem de script com código C. Além de se tratar da classe de linguagens mais popular atualmente para desenvolvimento multi-linguagem, características típicas de linguagens de script como coleta de lixo e tipagem dinâmica, por não estarem presentes em C, ilustram bem os problemas envolvendo a comunicação de dados entre diferentes ambientes de programação. Linguagens com tipagem estática podem apresentar necessidades de conversão de tipos semelhantes, mas o problema tende a ser simplificado pela definição de tipos equivalentes na API e inferência em tempo de compilação (vide exemplos nas API de Ada e Fortran com C). Linguagens funcionais possuem preocupações adicionais com efeitos colaterais no código C, mas isto é equivalente ao problema de quebra do paradigma causada pelo tratamento de entrada e saída que todas elas enfrentam.

Este estudo consiste de duas partes. Na primeira, realizamos a análise em profundidade de um conjunto de APIs para C providas por quatro linguagens de script – especificamente, Python (van Rossum 2006b), Perl (Wall 2000), Ruby (Thomas 2004), Lua (Ierusalimschy 2006) – além da provida pela linguagem Java (Gosling 2000). Diferentemente das demais, Java possui tipagem estática, mas assim como elas é baseada em um modelo de máquina virtual,

possui gerência automática de memória e permite carga dinâmica de código. Isto nos permite observar também como a tipagem afeta o projeto da API.

Na segunda parte, ilustramos as diferenças das APIs destas linguagens e o impacto destas no código resultante de uma aplicação C através de um estudo de caso. Realizamos uma comparação entre as APIs das linguagens de script através de um exemplo concreto, de modo a colocar implementações em cada uma das linguagens lado a lado.

O exemplo consiste de uma biblioteca genérica para scripting, chamada LibScript. Esta biblioteca foi projetada para tornar aplicações extensíveis através de scripting de uma forma independente de linguagem. Ela é baseada em uma arquitetura de *plugins*: bibliotecas dinâmicas carregadas sob demanda por LibScript que encapsulam a interface com as máquinas virtuais. A biblioteca principal provê uma API para scripting independente de linguagem, permitindo a uma aplicação registrar as suas funções e disparar scripts que utilizem estas funções. Esta biblioteca então invoca o plugin da linguagem apropriada para rodar o script (por exemplo, LibScript-Lua para código Lua). Assim, o código de cada plugin permite observar de forma clara e isolada os procedimentos adotados em cada linguagem para chamada de funções, registro de funções C e conversão de dados entre os ambientes.

1.2 Estrutura do texto

O trabalho está organizado da seguinte forma. No Capítulo 2, são discutidas as várias abordagens para interação entre código escrito em diferentes linguagens de programação. Partindo de uma visão geral, o foco se concentrará no método de interface externa mais comum nas linguagens da atualidade: interfaces com a linguagem C. Discutiremos os problemas comumente envolvidos na comunicação com código C e os modelos de programação que surgiram com a popularização da sua integração com linguagens de script. No Capítulo 3 são apresentadas em detalhe APIs para C de um conjunto de linguagens de script. Ao discutir estas interfaces, as diferentes soluções empregadas para os principais problemas envolvendo interação entre C e ambientes dinâmicos são levantadas. O Capítulo 4 exercita estas diferentes APIs através de um estudo de caso: uma biblioteca baseada em plugins que oferece uma interface uniforme simplificada para linguagens de script. Ao examinar a implementação de cada plugin, podemos comparar as APIs de cada linguagem realizando operações equivalentes. Finalmente, no Capítulo 5 são apresentadas as conclusões do trabalho, bem como apontados caminhos para trabalhos futuros.