AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,
INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

KATEDRA INFORMATYKI STOSOWANEJ

Praca dyplomowa magisterska

Realizacja podstawowej funkcjonalności maszyny wirtualnej Erlanga dla systemu FreeRTOS

Implementation of basic features of Erlang Virtual Machine for FreeRTOS

Autor: *Rafał Studnicki*
Kierunek studiów: *Informatyka*
Opiekun pracy: *dr inż. Piotr Matyasik*

Kraków, 2014

Oświadczam, świadomy(-a) odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

Spis treści

1. Język programowania Erlang

Niniejszy rozdział przybliża podstawy języka programowania Erlang. Zaprezentowany podzbiór języka został zaprezentowany w takim zakresie, aby możliwe było zrozumienie zadań, jakie do zrealizowania mają poszczególne elementy maszyny wirtualnej, zaimplementowane w niniejszej pracy. Szczegółowy opis języka został zawarty np. w pozycjach [1] i [2].

Erlang jest wieloparadygmatowym — funkcyjnym i współbieżnym językiem programowania ogólnego przeznaczenia o dynamicznym, lecz silnym typowaniu i automatycznym systemie zarządzania pamięcią. Motywacje, które przyczyniły się do zaprojektowania i implementacji języka oraz jego podstawowe założenia zostały przedstawione w podrozdziale **??**.

1.1. Typy danych

Język definiuje osiem podstawowych typów danych:

– liczby całkowite — operacje arytmetyczne na danych tego typu zapewnione są z nieograniczoną precyzją (ograniczoną tylko przez dostępną pamięć), dzięki wprowadzeniu do maszyny wirtualnej własnej arytmetyki stałoprzecinkowej. Przykładowymi wyrażeniami tego typu są `10` i `-25`;

– atomy — wyrażenia identyfikowane przez ciągi znaków zaczynające się małą literą lub zawarte w pojedynczych cudzysłowach, w maszynie wirtualnej są jednak zamieniane na liczbę całkowitą w celu szybszego porównywania ich. Przykładowymi atomami są `erlang` i `'EXIT'`;

– liczby zmiennoprzecinkowe — typ danych reprezentuje liczby rzeczywiste z 64-bitową precyzją, np. `3.14` czy `-2.718`;

– referencje — typ danych reprezentujący unikalne wyrażenie w zakresie klastra, służące do identyfikacji innych wyrazeń. Zmienna tego typu może zostać utworzona wyłącznie przez wywołanie funkcji `make_ref/0` wbudowanej w maszynę wirtualną;

– binaria — to ciągi bajtów zajmujących ciągły obszar pamięci, np. «`255,255,255,0`» czy «`ąbcd`»>);

– identyfikatory procesów — pozwalające na odniesienie się do wystartowanego procesu w maszynie wirtualnej poprzez wysłanie wiadomości lub zamknięcie go;

– porty — typ danych używany do komunikacji z systemem operacyjnym, np. systemem plików czy stosem sieciowym;

– lambdy — obiekty funkcyjne, które mogą zostać przekazane jako argument do funkcji wyższego rzędu i w niej wywołane.

Dodatkowo, zdefiniowane zostały dwa typy złożone:

– krotki — przechowujące określoną z góry liczbę innych wyrażeń (prostych lub złożonych). Dostęp do dowolnego obiektu w krotce możliwy jest w czasie stałym. Przykładem krotki jest `{salary, 100, 4.50}`);

– listy — przechowujące inne wyrażenia (proste lub złożone) na liście jednokierunkowej. Dostęp do dowolnego obiektu na liście możliwy jest w czasie liniowym. Przykładem listy jest `[salary, 100, 4.50]`.

Oprócz tego, język zapewnia dla „lukry składniowe”, które na etapie kompilacji kodu źródłowego zamieniane są na wymienione wcześniej typy danych:

– napisy — zapisywane jako ciąg znaków zawartych w podwójnych cudzysłowach, które zamieniane są na listę kodów ASCII poszczególnych znaków. Np. napis `"hello"` jest tak naprawdę listą postaci `[104,101,108,108,111]`;

– rekordy — pozwalające na odnoszenie się do poszczególnych pól krotki z użyciem nazwy (atomu), co upraszcza posługiwanie się tym typem danych.

1.2. Moduły i dynamiczna podmiana kodu

Jednostką kompilacji kodu źródłowego w języku Erlang jest pojedynczy moduł (plik z rozszerzeniem ***.erl**), którego proces kompilacji opisany został w dodatku **??**). Wszystkie skompilowane moduły nie są zależne od żadnych innych, dlatego procesowi kompilacji nie towarzyszy linkowanie modułów. Zależności pomiędzy modułami rozwiązywane są już w trakcie uruchomienia systemu, przez maszynę wirtualną.

Pojedynczy moduł składa się z zestawu funkcji: lokalnych i zewnętrznych. Funkcje lokalne możliwe są do użycia tylko i wyłącznie przez kod w danym module, natomiast funkcje zewnętrzne mogą być używane przez dowolny inny moduł. Dana funkcja jest funkcją zewnętrzną jeżeli została jawnie wyeksportowana z danego modułu poprzez użycie dyrektywy kompilatora `-export`.

Poszczególne funkcje rozróżniane są na podstawie: modułu w którym zostały zdefiniowane, nazwy oraz arności (liczby przyjmowanych argumentów). Na przykład, funkcja `bar` zdefiniowana w module `foo`, przyjmująca dwa argumenty oznaczana jest symbolem `foo:bar/2`.

Modularność implementowanych aplikacji pozwoliła na wprowadzenie do maszyny wirtualnej języka kolejnej ważnej cechy — możliwości dynamicznej podmiany kodu. Załadowanie nowej wersji danego modułu możliwe jest w każdej chwili uruchomienia maszyny wirtualnej. Nie ma to jednak wpływu

na wykonanie procesów korzystających ze starej wersji kodu, gdyż maszyna może przechowywać dwie różne. W momencie, gdy tej wersji modułu nie będzie wykorzystywał już żaden proces zostanie on usunięty z pamięci.

1.3. Kontrola przebiegu programu

1.4. Programowanie współbieżne i rozproszone

1.5. Obsługa błędów

1.6. Podsumowanie

Bibliografia

[1] Joe Armstrong. *Programming Erlang: Software for a Concurrent World. Second Edition*. The Pragmatic Bookshelf, 2013.

[2] Fred Hebert. *Learn You Some Erlang for Great Good!* No Starch Press, 2013.