GSM / Ю.А. Громаков // Электросвязь, 1993. – №10. – С. 9–12.
4. Алексеев Е.Р., Чеснокова О.В. GNU Octave для студентов и преподавателей. – Донецк.: ДонНТУ, Технопарк ДонТНУ УНИТЕХ, 2011. – 332с.

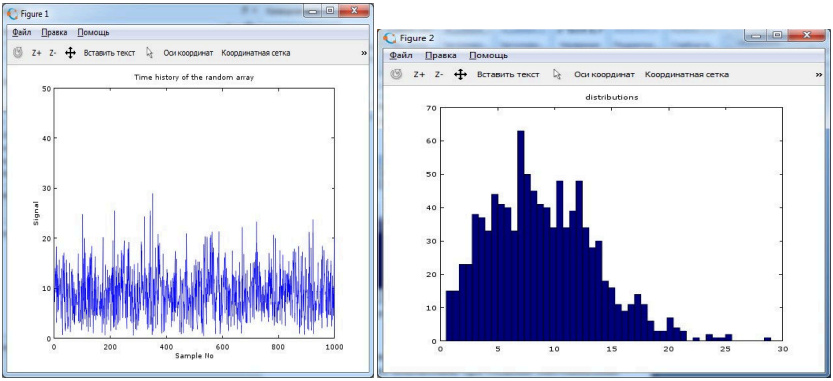

Figure 2 – Rayleigh noise and density histogram (Number of samples - 1000, Give the variance - 50)

Використання rust для реалізації розширень інтерпретованих мов

І. Лаврів

Lohika.Ltd, lavriv92@gmail.com

Rust is a general-purpose, multi-paradigm, compiled programming language sponsored by Mozilla Research. It is designed to be a "safe, concurrent, practical language",supporting pure-functional, imperative-procedural, and object-oriented styles.

Rust - це системна мова програмування, яка розробляється та підтримується організацією Mozilla Reasearch. Розробку мови почав у 2006 р. Грейдон Гоар, а в 2009 р. до розробки приєдналась Mozilla. Тривалий час Rust розроблявся, як експериментальна мова програмування і тільки в травні 2014 р. вийшла перша стабільна версія. Наразі стабільною є версія 1.8.0. Серед особливостей Rust можна виділити компіляцію з допомогою низькорівневої віртуальної машини, сфокусованість на безпеці та коректній роботі з пам'яттю, поєднання різних парадигм програмування.

Традиційна програма "Hello world" мовою Rust виглядає доволі просто і зрозуміло:

```
// hello_world.rsfn main() {  println!("Hello world!!!");}
```

У цьому прикладі за допомогою ключового слова fn оголошується вхідна функція, яка повинна мати ім'я main, і в тілі цієї функції за допомогою вбудованого макросу println! на екран виводиться повідомлення "Hello world".

Важливим елементом мови програмування є оголошення змінних, особливістю змінних в Rust є те, що вони за замовчуванням не змінюються та

є статично типізованими.

```
let a = 1; // за замовчуванням змінна незмінювана
let mut a = 1; // Ключове слово mut вказує, що значення надалі
    повинне змінитись.
let a: i32 = 2; // змінні можуть отримувати типи, як автоматично, так
    і явно
```

Ще однією цікавою особливістю Rust є алгебраїчні типи даних. Літера в назві типу вказує тип даних, а число їх довжину в діапазоні від 8 до 64 біт. Наприклад:

- i32 - ціле число довжиною 32 біти
- i64 - ціле число довжиною 64 біти
- f32 - число з плаваючою комою 32 біти

Мова дає змогу створювати структури даних за допомогою ключового слова struct

```
struct Point { x: i32,  y: i32}let point = Point {x:1, y:1};println!
    ("({}, {})", point.x, point.y); // виведе (1, 1)
```

Також можна описувати методи для обробки даних структури за допомогою ключового слова impl

```
impl Point { fn format(&self) -> &str { format!("({x.3}, {y.3})",
    self.x, self.y);}}
```

В Rust вбудована підтримка функціональної парадигми програмування.

- Оголошення функції.

fn sum(a: i32, b: i32) -> i32 { a + b}

- Підтримка лямбда-числення та функцій, як об'єктів першого класу.

let sum = |a: i32, b: i32| a + b;

- Підтримка функцій вищих порядків (Фунція може бути як аргументом, так і результатом іншої функції.)

```
fn test(n: i32) -> fn(i32) -> i32 {    let f = |a, b| a + b;
    return f;} let plus_one = |x| x + 1;test(plus_one);
```

Щодо абстракцій, то Rust дозволяє створювати трейти для даних та генерики для відділення опису та реалізації типів даних і функцій

```
struct Point<T> {x: T, y: T}let point = Point {x:1,y:1};let point1 =
    Point{x:1.0, y: 1.0 };
```

До репозиторію Rust також входить потужна утиліта для управління кодом проекту, вона дає змогу компілювати вихідний текст, запускати тести, збирати документацію, керувати сторонніми залежностями тощо.

cargo new project_name --bin # створює структуру для нового проекту cargo build # компілює код проекту cargo run # компіляція і запуск вихідного файла

Найпоширенішою нішею, де Rust знайшов вже своє застосування є написання системних утиліт та розширень для проектів, написаних іншими мовами програмування (python, ruby, javascript), та реалізація функціоналу недоступного цим мовам (швидкодія, багатопоточність, низькорівневе управління пам'яттю). Для інтеграції з цими мовами можна використовувати наступні способи:

- Реалізація мікросервісної архітектури з використанням черг

повідомлень;

- Виклик застосунків на Rust, як консольних утиліт;
- Взаємодія з допомогою FFI (Foreign functional interface).

Ось приклад написання модуля за допомогою FFI;

- Опис типу скомпільованого файла та залежностей проекту на Rust

```
# Cargo.toml[package]name = "test_addon"version = "0.0.1"authors =
    ["Ivan Lavriv <lavriv92gmail.com>"][lib]name =
    "simple_addon"crate_type = ["dylib"][dependencies]# List of your
    dependensies mongodb = "*"
```

- Реалізація модуля мовою Rust

```
// lib.rs #[no_mangle]fn public_function() {    // do something}
```

- Реалізація модуля обгортки (в цьому випадку мовою python).

```
# module.py import ctypesmodule =
    ctypes.cdll.LoadLibrary('./path_to_compiled_addon')mudule.public_
    method()
```

У разі реалізації розширення за допомогою FFI існує типова проблема синхронізації типів даних та семантичних особливостей між двома мовами. Для цього існують вже готові бібліотеки, які реалізовують особливості інших мов з допомогою особливостей Rust (rust-cpython, rust-rmi, тощо). Ось приклад використання rust-cpython

```
use std::any::Any;extern crate cpython;use
    cpython::ObjectProtocol;use cpython::{PyList, PyDict, PyObject,
    Python};#[no_mangle]fn get_data_as_dict() -> PyDict
    {}#[no_mangle]fn get_data_as_list(iterable: &[f64]) -> PyList
    {let python = Python(); let result = iterable.map(|&elem: &T|
    { match elem { Some(elem) => process(elem), None =>
    Python.None()}}); return PyList(python, result);}
```

Як висновок можна сказати, що Rust є сучасною, швидкою та безпечною мовою програмування, яка дає змогу програмісту цікаво і якісно вирішувати системні задачі в програмуванні.

Використання вільного програмного та апаратного забезпечення для точної реєстрації часових міток подій.

Мартинюк-Лотоцький К.П., Сергеев О.В.

1. Міжнародний центр Астрономічних і медико-екологічних досліджень, національна академія наук України; 2. Львівський національний університет імені Івана Франка, Астрономічна обсерваторія, langurek@gmail.com

Precise timing is one of the keystones for modern astronomical observations. The local timing service of the telescope Zeiss-600 at International Center for Astronomical, Medical and Ecological Research was upgraded. Its hardware consists of the GPS receiver and micro-PC RaspberryPi with RS422-to-TTL convertor. The corresponding software (Raspbian OS based) provides the necessary operations: timing data storage on SD card with access via ssh or ftp, providing local ntp server (it is synchronized over the public Internet and/or by PPS from GPS receiver). The developed ntp server maintains time to within 0.004 ms jitter