Proseminar "The Rust Programming Language"
Summer Term 2017

Garbage Collection & Reference Counting

Clemens Ruck & Alex Egger
Technische Universität München

July 12, 2017

Abstract

This paper aims to give a broad overview of existing memory management techniques and a quick introduction to the technique employed by the Rust programming language. Further it will demonstrate the different possibilities when working with memory in the Rust language.

1 Introduction

When talking about memory management one must firstly think about where in memory a given variable will be stored. There are different possibilities, that can be generally described by the following categories:

- Static memory
- Stack
- Heap

Statically allocated variables are simply hard-coded into the binary file of the program. The exact section they are located in is dependent upon the file format chosen.
Variables that are local to a called function are usually stored on the *stack*. The stack is a region in memory that emulates functionality of the stack data structure. It is simply a contiguous region of memory and a pointer to show where the last variable was placed. Since operating on the stack is such a common occurance, most CPUs have commands dedicated to pushing data to the stack and poping data from the stack, all while keeping the stack pointer at the correct location. This makes managing the stack trivial and fast.

2 Paper organization

The main part of a scientific paper is organized in a somewhat similar way each time:

1. You announce, that you will tell about something in the *abstract*
2. You introduce to what you did in the *introduction*
3. You clarify the amount of your specific contribution in the *related work*
4. You make the paper well-founded, by introducing necessary standards in the *basics part*
5. You actually tell about what you did in the *main part*
6. *Optional, if possible:* You evaluate in as neutral a way as possible what You have done in the *evaluation*
7. You summarize, and finally draw conclusions on what you have done in the *conclusion*

8. You sketch, what could be achieved next, based on this work in the *future work*

3 Basics

3.1 Garbage Collection

3.1.1 General principle

3.1.2 Reference Counting

3.2 Rust

4 Main

4.1 Heap allocation

4.2 Reference Counting in Rust

4.3 Raw pointers

5 Conclusion

6 Future work

Latex is the system of choice for scientific publications. Papers are expected to be delivered in Latex form, scientific publishers even require you to make use of their templates [4, 2]. We thus strongly encourage students to get to grips with Latex. As programmers, You will most likely enjoy the workflow of a text processor like Latex anyway.
The following is meant as a demonstration of the capabilities and the source code, that is necessary to produce this output. You may use the source code of this seminar paper as a template, if you wish.
First, we want to refer to the figures and the introduction. See Fig.**??** for the first floating figure with column width, and Fig.**??** for the one using the full page width. And here, we want to put a reference to the introduction which is Section 1.
In translating this template from German to English, I decided to stop here. There is not really much to get from the German text following. Anything Latex-related can also be looked up on the net. There is a *huge* number of tutorials, and so on.
Please do not use to much different font sizes and styles. It should be completely enough to go to *italic mode* for emphasizing something, such as newly introduced terms. You can refer to other parts of your paper (e.g. see Sec. 1). Quoting in Latex is done "this way". Further, you may have problems with punctation characters. Most of them just need to be prefixed by a backslash, for others you may temporarily switch to math mode: $ & %
{ } [ ] _ @ §< > \ @ ~/
Talking about math mode: you can do some very nice things this way:

$$a^2 + b^2 = c^2 \tag{1}$$

Again, refering to this equation is easy (see Eq. 1). If you do not need numbering for equations, use the *displaymath* environment:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Short equations simply can be used within the regular text flow, such as with $x \to \infty$. Obviously, math is fun with Latex.

Enumerations

Enumerations using bullet points:

- this is the first item of this list of interesting facts,
- second item,
- and the last one.

They also can be numbered:

1. item one,
2. item two,
3. item three.

As shown, numbers always should be written out in the text, unless the belong to a title or a formula.

7 Literature

At the end of your paper, you should have a nice list of used literature. For scientific papers, this actually is needed. You always use other works as base for your own. Usually, you are not the only one thinking about a given difficult problem, so there is always related work which *must* be cited if known to the author.
Further, if you want to copy relevant sentences from an original paper, you *have* to cite them correctly, for example in this way:

> "I think there is a world market for maybe five computers." (T.J. Watson, IBM, 1943)

However, in computer science, direct citations are uncommon if not even considered bad style – with the notable exception of lemmas and theorems.
Much more common is the indirect citation. Here, the cited work (especially all the regular text) must be written/phrased by you. If you write about some results or fact stated in another paper, you should refer to it. The 'Analytical Engine" — a mechanical calculation machine — created by Charles Babbage in the year 1838 was based on the decimal system [5].
You can find new sources for scientific topics quite comforatbly via DBLP Computer Science Bibliography [3]. This site not only acts as a registry for almost all publications in computer scientific journals and conferences, but also provides You with correctly formated *bibtex*-entries, ready to be integrated into your seminar papers *bib*-file.

8 Figures and Tables

No need to understand the following text.
Figures can span either one column (see Fig. **??**) or the full page width (see Fig. **??**). Latex automatically tries to find the best place for these floating figures. To influence that, you may move the figure a bit to the front of your text. As can be seen in Fig. **??**, using raster images usually results in quite bad quality. Better use vector formats: draw the figures with *inkscape* [1], and save them as PDF or SVG. As example of this procedure, see Fig. **??**).
Similar to figures, tables can be referred to in the text (see Tab. 8). However, sometimes it is useful to embed tables directly in the regular text flow:

	Column 1	Column 2
Row 1 Row 1		

9 Summary

The summary shortly repeats the core ideas and results from the previous text. If the reader has problems understanding the summary he knows that he should go back to the relevant sections. Thus, the last section should consist of:

- a summary,
- an evaluation of what was done, importance of this work,
- what is left, what still needs to be done,
- short outlook into the future.

Last but not least, we can explain anything missing yet in the evaluation done in this paper. This allows to refer to what readers can expect from authors in the future.

References

[1] Inkscape – Draw Freely, 0.92.1. http://inkscape.org.

[2] 2017 ACM Master Article Template, 2017. https://www.acm.org/publications/proceedings-template.

[3] DBLP Computer Science Bibliography, 2017. http://dblp.uni-trier.de.

[4] Information for Authors of Springer Computer Science Proceedings, 2017. http://www.springer.com/de/it-informatik/lncs/conference-proceedings-guidelines.

	Column 1	Column 2	Column 3	Column 4	Column 5	Column 6	Amount
Row 1	This column has a maximal width of 2 cm.	X	X	X	X	X	126,00
Row 2		This entry occupies three columns.			X	X	8,00
Sum							134,00

Table 1: For this layout, we want table captions to be *below* the actual table.

[5] A.G. Bromley. Charles Babbage's Analytical Engine, 1838. *Annals of Computer History, IEEE*, 20:29–45, 1998.