
Fakultät Informatik

# Development Report about Ion shell for Redox Os

It Project Report im Studiengang Informatik

vorgelegt von

Florian Naumann

Matrikelnummer 3528558

Betreuer: Prof. Dr. Christian Schiedermeier

# Contents

## 0.1 Abstract

This report describes the development contribution to Ion Shell for the It project "redox" between the winter term 2022 and summer term 2023. First an introduction about Ion Shell is provided. In the next chapter the typical development process which I applied, is shown via examples of bug fixes and feature implementations for Ion Shell. In additions that section indents to showcase how documentation, testing and coding can be performed for Ion Shell. After that chapter a reflection outlines the current status of Ion Shell. This reflection also involves achievements of my development work and left tasks/goals I could not complete. In the last chapter a listing of all my created issues and conducted pull requests are provided to quantify the development work.

## 0.2 Abstract in German

Dieser Bericht beschreibt die Entwicklung als Beitrag zu Ion Shell für das It Projekt "redox" während dem Wintersemester 2022 und Sommersemester 2023. Zuerst kommt eine Einleitung zu Ion Shell. Im nächsten Kapitel wird der typische Entwicklungsprozess, den Ich angewendet habe, durch Beispiel in Form von Fehlerbehebung und Implementierung von Features aufgezeigt. Zusätzlich soll dieses Kapitel demonstrieren wie Dokumentation, das Testen und die Programmierung für Ion Shell durchgeführt werden kann. Nach diesem Kapitel soll der aktuelle Status von Ion Shell verdeutlicht werden. Dieser Teil soll auch die Errungenschaften meiner Entwicklungsarbeit und die übrigen Aufgaben/Ziele welche Ich nicht vollendend konnte, aufzeigen. Im letzten Kapitel wird eine Auflistung von all meinen erstellten Issues und durchgeführten Pull Requests dargestellt um meine Entwicklungsarbeit zu quantifizieren.

## 0.3 Sources of code snippets

In several places, code snippets are used to illustrate certain points. Code snippets are taken from the Ion Shell repository on Gitlab [1]

based on the commit id 60bfb73351f0412c95b8ba2afe75e988514470a6. Unfortunately I could not generate an online link to a file based on a certain commit. For this reason I include a local copy of the repository beside this report. The name of this snapshot is "ion.zip". This local copy is a snapshot of the repository based on the commit 60bfb73351f0412c95b8ba2afe75e988514470a6. A source item in the bibliography of a code snippet is therefore a local file path within this local snapshot.

The Notation for file paths is as follows:

“‘ <folder> -> <folder> -> .. -> <file> “‘

Therefore the following Example:

“‘ src -> shell -> flow.rs “‘

Equals the flow.rs file inside the folder shell which is inside the folder src at the root project.

# Chapter 1

# What is Ion Shell

Ion Shell is a program written in rust. It is maintained on a remote git repository on the GitLab server for Redox ecosystem [2]. It is a shell which executes commands within a terminal emulator via a read evaluation loop. Like other shells, Ion Shell also allows the execution of scripts in its own language. It serves as the default shell on redox os. As moment of writing ion shell can also be used on Linux. This shell is interpreted statement by statement. There is an online manual for the usage of ion shell available [3].

The work which was performed during the it project resolved mainly around the scripting aspect of Ion Shell. Because of this the scripting language of Ion Shell is introduced in more details as a preamble. To better illustrate certain aspect of scripting with this shell, certain differences between It and bash are discussed.

## 1.1 Typing in Ion Shell

As an example Bash only operates on text and does have not have concept of types. Ion Shell on the other hand works in structural typing. A certain section of the online manual of Ion Shell lists all possible types [4]. Form this reference we conclude that Ion shell has following primitive types:

- str
- bool
- int
- float

And also provides these structural typing via:

- Array

- Hashmap
- Btreemap

Working with these types instead of just text provides a better detection of programming errors. Since Ion Shell is interpreted statement by statement, these errors are caught at runtime though.

It is important to note there is no nominal typing in Ion Shell. Formal Declaration of objects or structs with named fields is not possible like in other programming languages like Rust, Java, C# etc ..

To better illustrate how scripting in Ion Shell looks like, a few core features are explained with code snippet as examples in next subsections within this chapter.

## 1.2 Expansion

Ion Shell also supports expansion functionallity. This features allows to replace text with other values. or other expression with Most of time expansions are prepended with a sigil. A sigil is a character which signifies which type a resovled value is of. For instance this allows resolve the variable name to its respective value, see code snippet on 1.1 at the line 3. Here the sigil "$" is used to expand a reference to a string value. At line 11 in the snippet 1.1 there is a showcase of the expansion of a variable to an array via the sigil "@". These two examples demonstrates that Ion Shell differenciate between strings and array. The brace expansion feature is illustrated at line within the snippet. Ion Shell enables conducting math via the arthimetic expansions, starting from line 36 in the code listing 1.1 shows how arthimetic is performed.

```
1  # String expansion
2  let string = "example string"
3  echo $string
4  echo $string:$string
5  # Output
6  # example string
7  # example string:example string
8
9  # Arrays are with an different sigil expaneded "@" instead of "$"
10 let array = [one two three]
11 echo @array
12 # Output
13 # one two three
14
15 # Brace expansion with different types
16 echo {1..10}
17 echo {10..1}
18 echo {1...10}
19 echo {10...1}
20 echo {a..d}
21 echo {d..a}
22 echo {a...d}
23 echo {d...a}
24 # Output
25 # 1 2 3 4 5 6 7 8 9
26 # 10 9 8 7 6 5 4 3 2
27 # 1 2 3 4 5 6 7 8 9 10
```

```
28 # 10 9 8 7 6 5 4 3 2 1
29 # a b c
30 # d c b
31 # a b c d
32 # d c b a
33
34 # Arithmetic Expansion
35 # Addtion
36 echo $((a + b))
37 # Raising to power of 2
38 echo $((a ** 2))
```

Listing 1.1: Expansion of variables. Source: [5].

## 1.3 String and array methods

There are builtin utility functions avaible for scripting called methods. According to the Ion manual methods are a subset of the expansion feature of Ion Shell. There are 2 kinds of methods. The kind of method is determined by the type of the return value. Every kind of method is prepended with its own sigil.

- String methods. Prepended with a "$" sigil. Returns a string. See 1.2 as an example.
- Array methods. Prepended with a "@" sigil. Returns an array. See 1.3 as an example.

```
1  # Lines splits a string into an array of elements.
2  # These elements are seperated by "\n" as new line.
3  echo @lines($unescape("firstline\nsecondline"))
4  for line in @lines($unescape("third\nfourth\nfifth"))
5      echo $line
6  end
7  # Output
8  # firstline secondline
9  # third
10 # fourth
11 # fifth
```

Listing 1.2: Example of a string method. Source: [6].

```
1  # Returns a string which the repetition of the 1. argument
2  # times 2. argument as a number.
3  echo $repeat("abc, " 3)
4  # Output:
5  # abc, abc, abc,
```

Listing 1.3: Example of a array method. Source: [7].

## 1.4 Control Flow

Ion Shell offers control flows to give conditional and repeated execution of statements. The syntax of control flows resembles the syntax of Rust to a certain degree. There are three kinds of control flow:

- Conditional execution by if statement. Example: 1.4.
- Repeated execution via while or for loops Example: 1.5.
- Value matching through matches statements. Example: 1.6.

```
let foo = "foo"
if test "foo" = $foo
    echo "Found␣foo"
else if matches $foo '[A–Ma–m]\w+'
    echo "we␣found␣a␣word␣that␣starts␣with␣A–M"
    if not matches $foo '[A]'
        echo "The␣word␣doesn't␣start␣with␣A"
    else
        echo "The␣word␣starts␣with␣'A'"
    end
else
    echo "Incompatible␣word␣found"
end
```

Listing 1.4: Showcase of if statements in the Ion Shell. Source: [8].

```
let value = "one␣two␣three␣four"
for element in @split(value)
    echo $element
end
let value = 0
while test $value -lt 6
    echo $value
    let value += 1
end
```

Listing 1.5: Showcase of loop in the Ion Shell. Source: [9].

```
let bar = "bar"
let foo = bar
match $string
```

```
4       # if guard allows for more complex conditional logic within matching valu
5       case "this" if eq $foo bar
6           echo "this␣and␣foo␣=␣bar"
7       case "this"
8           echo "this␣and␣foo␣!=␣bar"
9       case _; echo "no␣match␣found"
10  end
```

Listing 1.6: Showcase of switch matches in the Ion Shell. Source: [10].

# Chapter 2

# Development in Ion Shell

## 2.1 Reading up on and becoming acquainted with Ion Shell

Before contributing, knowledge and a certain understanding of Ion Shell and Redox Ox needed to be obtained. The following activities were undertaken to get ready for development in Ion Shell:

- Reading the Redox Os book [11].
- Going through the online manual of Ion Shell [12].
- Reading posted issues on the GitLab Repository of Ion shell.
- Reading the contribution guideline of Ion Shell [13].
- Explore scripting of Ion Shell.

The online manual of the shell is meant for the users of the application. It focuses on the following points:

- Features of the scripting language of Ion Shell.
- What and how something can be done in an interactive session.
- Philosophy the application. Why does it exist and what it is supposed to be.

Ion manual is build by the program of the rust program mdBook, [14], which converts markdown file into folder of HTML, CSS and JavaScript files for hosting a documentation as a website. Investigating the online manual of Ion shell provided the overview of already implemented features and how the shell was meant to be used. While reading the manual, it became clear that the documentation lacked explanations in certain areas. Later on additions to the documentation were also required for new implemented features in Ion Shell. Ion shell's manual inevitably became one of the fields for contribution. Another

important source for the deduction of tasks was the collection of issues and pending pull requests from Ion shell's repository. Over the course of the it project general knowledge about Redox OS was extracted from the official redox OS book [11]. The good practices for development were derived from the Contributing markdown file of the Ion Shell repository [13].

## 2.2 Development in general

The whole program is developed in the Rust programming language like the whole ecosystem of Redox os. Beside the Rust, knowledge about Make files proved invaluable to understand the working of the integration tests and allowed to add more capability of running only one specific interrogation test, accomplished by the pull request [15]. The mastering of the scripting language of Ion Shell was vital too. Without it I could have not found out the correct behaviour which is to be achieved after a bug fix or feature implementation.

The communication with redox OS community were done by a matrix sever [16]. As contributor to Ion Shell I forked the Ion Shell repository and developed on this fork. Once a change to the code base was ready to be merged, I prepared the pull request, according to the official redox OS guideline about pull requests [17].

## 2.3 Verification of bug fixes and implementation via automated testing of features

The main way to verify correctness of features and bug fixes is the usage of unit tests and integration tests. Unit tests and integration tests are the 2 forms of tests used to verify the functionality of the shell. The observation of the code base suggests that the first form as unit tests are mainly used to inspect the correctness of tokenisation, parsing and evaluation of the builtin functions of the shell in isolation. It is important to note that these unit tests execute and assert inner/private functions too in contrast to other programming languages like CSharp where unit test should not access private functionality easily. This is typical in the Rust language because unit tests are often in the children module in respect of the code to be tested. Visibility in Rust enables code in a children module to have full access to all functionality in a parent module. This allows to test private functions in Rust easily [18].

Integration tests depend on the kind of application. Ion shell's approach to integration test is the execution of script files and comparing the output to text files which contain

the expected output for the test. This kind of integration test therefore focuses on the scripting aspect of the shell. Especially this kind of testing is valued highly according to the Contributing Guidelines of this project [13]. Since rust does not provide a builtin way to perform these kinds of tests, a rather complex bash script was written to orchestrate this kind of integration tests. Before this it project there was no explanation of how these new integration tests are executed and how they should be written. It was however paramount to add new integration tests to verify bug fixes and implemented features. A documentation section in the contributing guideline was crafted during the process of learning the working of this integration tests [13]. This will hopefully reduce the learning effort for future contributors in the area of integration tests.

## 2.4 Example of development

### 2.4.1 Implementing an array method in Ion Shell

The implementation of the feature of the method "subst" ought to represent in the development work in Ion Shell. This method allows to provide a default value for an empty array. The following tasks were performed to archive the implementation:

- Determine the desired behaviour of the array method.
- Create a test to verify the functionality of the implementation.
- Implement the method in Ion shell.
- Document this feature in the online manual of Ion shell.

In the end the implementation was proposed and merged under the following pull request [19].

#### 2.4.1.1 Finding out the desired behaviour of "subst"

This feature was requested in an issue a certain while ago [20]. In this issue the specifics of behaviour of this array method have already been discussed too. The discussion concluded the following signature of "subst".

“‘text subst(input: T, default: T) -> T “‘ source: [21]

Where the invariant says that T must be of type array. This array method takes two arguments as arrays and returns either the first or second argument. The first one is returned if it is not an empty array. Otherwise the second argument is returned. The accomplished implementation follows the signature and invariant.

### 2.4.2 Creating integration test for the implementation

To verify the functionality of the implementation, an integration test was added to test suit. An integration test consists of 2 files. One is an ion file which is executed by the shell an other is a plain text file with extension "out". This text file describes the expected output of the execution of the ion test file. For this implementation there is the file "subst.ion" and "subst.out" where "subst" is the name of the test. Both file must share the same base file name, here "subst".

The code snippet 2.1 represents the test code and should produce the following expected output 2.2.

In additions to the integration tests unit tests were put in place to verify the finer details of the implementation. Every unit test is presented as function annotated with "#[test]" in the code snippet 2.3.

```
let array = []

# Inline array in the method
echo @subst(@array [foo bar])

# single value
echo @subst(@array [baz])

# variable expansion
let default = [foobar]
echo @subst(@array @default)

# method would not trigger
let array ++= faz
echo @subst(@array @default)

for number in  @subst([] [2 3])
  echo $number
end
```

Listing 2.1: Ion shell script for testing implementation of subst method. Source: [22].

```
foo bar
baz
foobar
faz
```

```
2
3
```

Listing 2.2: Expected output of executed test script for subst 2.1.. Source: [23].

```
...
#[test]
fn test_subst_variable_over_default() {
    let method = ArrayMethod::new(
        "subst",
        "@ARRAY",
        Pattern::StringPattern("[2 3]"),
        None
    );
    assert_eq!(
      method.handle_as_array(&mut DummyExpander).unwrap(),
      args!["a", "b", "c"]
    );
}
#[test]
fn test_subst_default_over_empty_array() {
    let method = ArrayMethod::new(
        "subst", "[]", Pattern::StringPattern("[2 3]"),
        None
    );
    assert_eq!(
      method.handle_as_array(&mut DummyExpander).unwrap(),
      args!["2", "3"]
    );
}
#[test]
fn test_subst_fail_no_array_variable() {
    let method =
        ArrayMethod::new(
        "subst",
        "not an array",
        Pattern::StringPattern("[2 3]"),
        None
    );
    assert!(method.handle_as_array(&mut DummyExpander).is_err());
}
```

```
37 #[test]
38 fn test_subst_fail_default_is_no_array() {
39         let method = ArrayMethod::new(
40         "subst", "[]",
41         Pattern::StringPattern("not an array"),
42         None
43     );
44     assert!(method.handle_as_array(&mut DummyExpander).is_err());
45 }
46 #[test]
47 fn test_subst_fail_default_missing() {
48         let method = ArrayMethod::new(
49         "subst", "[]",
50         Pattern::Whitespace,
51         None
52     );
53     assert!(method.handle_as_array(&mut DummyExpander).is_err());
54 }
55 ...
```

Listing 2.3: Unit test for the implementation of the method "subst".. Source: [19].

### 2.4.3 Implementation of "subst"

After settling goals and the verification of the implementation it is time make this feature a reality via Rust code. The implementation of the method is contained the function shown in 2.4. The first part of the function validates if the first and second argument are of type array, from line 4 to line 28. This validation ensures that an error is returned if the user violates the conditions which were lay out, see 2.4.1.1 section. Errors in Rust are propagated by value rather than exceptions. This is done via the generic enum type Result<T, E> where T is the type of the value in the successful case and E is the type of value which communicates what went wrong in case of error. A functions therefore declares via its return type of the enum Result<T, E> that it can fail, line 2. Within the code section for validating the arguments, the function returns Err variants, case of the violation of the stated conditions 2.4.1.1, at several places, at line 5, 14 and 23. There is another spot where an error is returned. This spot can be found by the "?" character which signifies that if an error is given, then propagate this error to the caller of the function at line 28. This propagation equals a rethrowing of an exception in C#:

```
try {
```

```
var array = expand_func.slice_array(elements, this.selection);
} catch(Exception e) {
throw e;
}
```

Between line 34 and line 46 the success case is performed by returning the first argument back if it is not empty, line 45. The second one is returned only if the first one is an empty array, line 43.

```
1  fn subst<E: Expander>(&self, expand_func: &mut E)
2      -> Result<Args, Error<E::Error>> {
3      let variable = self.resolve_var(expand_func)?;
4      if !is_array_expression(&self.variable) {
5          return Err(MethodError::WrongArgument(
6              stringify!(subst),
7              "1. argument must be an array",
8          )
9          .into());
10     }
11     let default_over_empty = match self.pattern {
12         Pattern::StringPattern(pattern) => {
13             if !is_array_expression(pattern) {
14                 return Err(MethodError::WrongArgument(
15                     stringify!(subst),
16                     "2. argument must be an array",
17                 )
18                 .into());
19             }
20             Self::resolve_arg_array(expand_func, pattern)
21         }
22         Pattern::Whitespace => {
23             Err(MethodError::WrongArgument(
24                     stringify!(subst),
25                   "requires a 2. argument").into()
26               )
27         }
28     }?;
29     let elements = variable.split(char::is_whitespace);
30     let array = expand_func.slice_array(
31       elements,
32       &self.selection
```

```
33      )?;
34      if array.is_empty()
35          // returned slice_array array can still
36          // have one element with nothing it even the
37          // variable was an empty array.
38          || array.get(0)
39            .expect(
40                "Unexpected: array was checked if it was empty"
41              ).is_empty()
42      {
43          Ok(default_over_empty)
44      } else {
45          Ok(array)
46      }
47  }
```

Listing 2.4: Function which implements the subst method. Source: [19].

### 2.4.4 Documentation of "subst"

After the implementation an explanation of this method was noted down in array method section of the Ion Shell manual [7]. This documentation covers the following aspects:

1. What does it do.
2. What is expected of the provided arguments and when an error is raised.
3. Example of the usage with an expected output.

The first and second point are communicating the established conditions the functions expects to be called, shown at 2.4.1.1.

# Chapter 3

# Conclusions after the development

## 3.1 Accomplishments

Over the period of the it project, participation in a number of areas of development were performed:

- Improving Code quality for example refactoring.
- Documentation.
- Writing unit tests and integration tests.
- Bug fixes.
- Implementation of features.

It was possible to get the code quality to a point where no warnings or complication errors occurred. The all pull requests described in the section 4.1.3 contributed to this success.

Contribution in regard of documentation was another portion of development, listed in 4.1.4. This not involved the markdown files at the project root and the comments within the source code but also the part of the ion manual. This will hopefully server future contributors well.

Ion Shell as any other software contains bugs of course. Some the bugs could be fixed via code contribution on my side, listed in 4.1.2. A few features were also achieved, listed in 4.1.1.

The integration tests of Ion Shell were analyzed and comprehended. On one hand this knowledge could be leveraged to create effective tests which verify the correct behaviour of the implemented features and bug fixes. On the other hand this understanding lead to a section about how to write the integration tests in Ion Shell [24].

## 3.2 Status of Ion Shell

### 3.2.1 Capabilities of Ion Shell

Ion Shell is not only capable of running on redox os but also quit well on a Linux distribution. This fact became more than clear by the my conducted development of Ion Shell on a Linux distribution.

The scripting API of Ion Shell allows for writing more correct automation compared to shells like bash shell. Structural typing and various higher level language features like expansion 1.2 methods 1.3 and control flow 1.4.

### 3.2.2 Problems with Ion Shell

However Ion Shell has unfortunately several troublesome areas which hinders development and wider adoption among users. This section points out those aspects of Ion Shell which should be improved on.

- Ion Shell has no regular contributors
- Ion Shell CI is not functioning correctly
- Ion Shell has no distribution for end user on Linux
- Ion Shell has not mainted dependencies

### 3.2.3 Ion Shell has no regular contributors

The repository of Ion Shell does not indicate an active development community from the beginning of the it project up to the moment of writing. Before my first contribution the last commit of someone else than the BDFL of Redox Os, Jeremy Soller, was on Nov 27, 2022. Up to the moment of writing most commits were done by me. An exceptions was an occasional pull request [25]. Throughout the it-project there was only one person, BDFL of Redox Os, who reviewed and accepted pull requests. A prolonged lack of regular contributors could lead to a complete halt in the development of Ion Shell.

### 3.2.4 Ion Shell CI is not functioning correctly

A number of dependencies used by Ion Shell are not maintained anymore ... If newly needed features or bugs arise from those libraries, then someone has to do it on their own. This puts more complexity on the new potential contributors as they might have to learn and adjust separate code bases.

### 3.2.5 Ion Shell has no distribution for end user on Linux

Ion Shell can be still considered an alpha program. During the development phase of it project I discovered pretty fast new bugs and needed features. With a larger user space these issues could be detected early. More users could provide more feedback to the program and even attract potential contributor. Additional contributors is what this project really needs the most. Ion Shell can be complied and run on Linux. However it lacks official distribution channels for linux end user . A user indenting to use this shell, has to install the rust tool chain before and then compile the program. More Linux use rend users could be reached the program would be released on distribution channels like dnf, apt etc .. .

### 3.2.6 Ion Shell has not mainted dependencies

Not all jobs of the Ion Shell CI' do still not complete successfully. This is problematic. While local compilation and manually run test suit can provide detection of problems, it not enforced. Every pull requests therefore require also inspection of the compilation and a working test suit for the code reviewer. This could be automated with a proper CI. Beside causing more work for the reviewer, it also can hinder the confidence of new contributors not knowing that CI can run successfully right now. I tried communicating this problem via following issues [26]. In addition I talked to the community about this on the matrix server on several occasions [16].

## 3.3 What is left to do in Ion Shell

This section describes which goals for the it project could not be archived. The description also contains the reasons why a goal could be accomplished.

- Incorporate the source location, (file name, line number, column number) into the error message.
- Implement wild card expansion for various schemes like disk ..

- Make the CI work on all jobs possible.

The following bullet points were non-goals during the it-project. Nonetheless they were identified as worthwhile endeavours.

- Provide various deploy set ups to publish Ion Shell on various package managers like apt, dnf, ..
- Port Ion Shell to windows.

### 3.3.1 Enriching error messages

One area of contribution was focused on enriching error message in Ion Shell. Error messages in Ion Shell already provide specific information about what problem occured. However the details within certain error messages had some potential for improvement. The following merged pull requests of mine were focused on that areas:

- Out of bounds errors show the invalid index and the length of an array [27].
- Errors because of the reference of an undeclared variable now show the name of the undeclared one [28].

Another aspect of missing information for error messages was the source location where an error was encountered. The source location is the respective line and column number. The source location in an error message as a feature were noted down in an still opened issue [29] . It was my indentation to add the source location to a raised error message. In the end of the it project I was not able to do so though. The parser of ion shell does not keep track of line and columns numbers for statements. Because of this, one can not simply incorporate the source location into the error messages.

A workaround was attempted via counting at least the line numbers per executed statement. There is a for loop, see code snippet 3.1 as a simplfied and modified version, which terminates, parses and executes statements one by one in Ion Shell. In the modified version there two additional statements. The first addition the declaration of the line numbers to count per statement at line 9. The second addition the increment of the line numbers per statement at line 18. Those modifications ought to keep track of the line number of the current statement.

```
1  /// Receives a command and attempts to execute the contents.
2  pub fn on_command(
3      &mut self,
4      command_to_execute: impl Iterator<Item = u8>,
5      set_cmd_duration: bool,
6  ) -> std::result::Result<(), IonError> {
7      // ... some code before
8      // Added by workaround
9      let mut line_number: usize = 0;
10     for stmt in command_to_execute.batching(
11         |cmd|
12         Terminator::new(cmd).terminate()) {
13         // Go through all of the statements and build up
14         // the block stack
15         // When block is done return statement for execution.
16         for statement in StatementSplitter::new(&stmt) {
17             // Added by workaround
18             line_number += 1;
19             let statement = parse_and_validate(statement?)?;
20             if let Some(stm) = Self::insert_statement(
21                     &mut self.flow_control, statement
22                 )? {
23                     self.execute_statement(&stm)?;
24             }
25         }
26     }
27     // ... some code after
28     Ok(())
29 }
```

Listing 3.1: For loop which terminates,parses and interprets statements. Source: [30].

That attempt however did not work for all scenarios. In cases of calling a function the line numbers were not correct however. This oddity comes from the fact that statements within a function are executed in a batch. Let us consider the following ion script snippet 3.2. If an error occurs at line 3, then the line number is wrongly 6 with the workaround because counted line number refers to the function call.

```
fn print
  let message = "2"
  echo $massage
end

print
```

Listing 3.2: Example with undefined variable error.

In addition the execution of string and array methods do not propagate their error up to the execution loop 3.1 like the rest of the code base. If an error is caused, a message is directly printed to the console, between line 13 and 21 within 3.3. This not consistent and should be refactored so that the error is propagated to the previously mentioned execution loop 3.1.

```
/// The exit status of a command
///
/// Provides some helpers for defining
/// builtins like error messages and semantic constants
#[derive(Clone, Copy, PartialEq, Eq, Hash, Debug, Default)]
pub struct Status(i32);
impl Status {
    // ... other declarations
    /// A generic error occured. Prints an helper text
    pub fn error<T: AsRef<str>>(err: T) -> Self {
        let err = err.as_ref();
        if !err.is_empty() {
            eprintln!("{}", err);
        }
        Self(1)
    }
    /// Wrong arguments submitted to the builtin
    pub fn bad_argument<T: AsRef<str>>(err: T) -> Self {
        let err = err.as_ref();
        if !err.is_empty() {
            eprintln!("{}", err);
        }
        Self(2)
    // ... other declarations
}
```

Listing 3.3: How builtin method print raised errors directly to the consoel. Source: [31].

In my view the best solution is to rewrite the tokenisation and parsing. This rewrite should make token/language items linked to their respective column and line number. That refactor however requires coordination with the repository owners and significant amount of work which exceeds the available time I could afford to spend for the it project.

### 3.3.2 Wild card expansion of schemes resources

Since Ion Shell is meant to be used as the default shell within Redox os, the correct handling of paths to resources of schemes is a vital part. Schemes in redox Os facilitate inter process communication on redox Os. A scheme signifies a certain type of a resource. The following snippet shows an example of reference to a resource within the file scheme.

```
file:/home/some_user/some data
```

The name of the scheme comes before ":" character. The part behind that character resembles a reference to a resource, a file or directory with the name "some data".

Ion shell as program running in Redox Os, has the kernel handle the resolution of resources of certain scheme. However the kernel itself does not have the concept of wild cards within a reference of scheme. According the conses among the Redox Os developer community here Ion Shell is meant to jump in and handle the resolution of wildcards. This opinion came up through a discussion on the matrix server, [16]. The following example is expected to be resolved to all paths under the folder "/home/some_user".

```
file:/home/some_user/*
```

While developing on Ion Shell I could implement a workaround for the wild card expansion of the file scheme via the pull request [32] . This pull requests relates to the issue [33]. Expansion for other schemes like "disk" are still missing. This missing feature is documented by following issue [34]. Over time it became clear that the wild card expansion for schemes should be implemented in separate rust crate, within the Redox Os community. Creating a separate library is outside the scope of this it project though.

### 3.3.3 Make all jobs work on the CI

As described in one of the improvable development aspects of Ion Shell, 3.2.2, the CI of Ion Shell should work properly. The job 'linux:stable' still fails. Mending this job of the CI probably requires contribution on redoxer repository [35] and an update of the docker image on docker hub [36].

### 3.3.4 Make Ion Shell available as binary on different Linux distribution

At the moment of writing this report, the only way to use Ion Shell as an end user is to compile the program with the rust compiler. The problem caused by that are described in 3.2.6.

### 3.3.5 Porting to windows

According to a page of Ion manual a port to windows is desired [12]. This port could indeed increase the number of users since windows as an OS has a large user base in the desktop realm.

### 3.3.6 Migration from Makefiles/Bash to cargo-xtask automation

While cargo for Rust simplifies compiling and running of rust programs, task like running the test suit or building the ion manual are however project specific. Specific project task are done by Bash scripts and Make files within the Ion Shell repository. As pointed out in a posted issue [37], this approach does not scale well on the long run. This issues proposed the usage of the cargo-xtask convention instead of Make files and bash scripts. Cargo-xtask is a convention in which the workspace feature of cargo is utilized [38]. Via this workflow it is possible to write project tasks in Rust. The following advantages arise from this convention:

- Rust scales better for complex problems compared to Bash.
- Only one language in the project for the application and for the automation system.
- People do not need to learn another stack beside Rust.
- Rust has powerful crates for cli argument parsing. Their Capabilities out match getopts from bash greatly
- It is advantageous to get rid of make and bash for a latter windows port.

After the discussion on the matrix server with community of Redox OS and under the respective issue [37], it was agreed to try the conversion. The conversion to cargo-xtask was started with the following pull request [39]. This pull request is the first of three planned pull requests for the conversion. That strategy allows to conduct the conversion step by step However the first pull request is not merged yet, even after asking for a review on this pull request several times.

# Chapter 4

# List of development accomplishment

## 4.1 Pull Requests

### 4.1.1 Features

- Implemented pipefail feature. If at least one command in pipeline returns an error, the whole pipeline returns an error.[40].
- Improved error message for not declared variable. Now shows the name of not found variable [28].
- Improved error message for invalid index. Message now shows the length of sequence where the out of bounds error occurs [27].
- Implemented work around to make file scheme work in Ion Shell inside Redox Os [32]
- Implemented subst method for Ion Shell. [19]

### 4.1.2 Bug fixes

- Make all buitlins of ion shell included the ion manual after building documentation [41].
- Ensured that spaces between left parentheses and the first argument does not cause an error anymore .[42]
- Converted unguarded unsafe conversion to utf-8 string. [43]

Related to redoxer

- Fixed semantic version of redoxer and applied missing argument for a function to fix compiler failure [44].

### 4.1.3 Code Quality

- Scripted option to only run one selected integration test. [15]
- Removed redundant, more than once, termination before tokenisation. [45]
- Remove obsolete parameter in several functions in the parser section of Ion Shell. [46]
- Fixing warnings by using modern API of criterion in benchmarks. [47]
- Added unit tests for parsing let statements. [48]
- Added unit tests for terminating lines before tokenisation. [49]
- Removed not used struct field. [50]

### 4.1.4 Documentation

- Moved msrv from clippy file to Cargo.toml. [51]
- Documented how the documentation of builtin are generated into the Ion Shell online manual. [52]
- Wrote section about how integration test work and how to write them. [53]
- Provided additional command for building ion manual locally. [54]
- Fixed formatting in the builtin section of the ion manual [55]
- Adjusted msrv to sync it within the project [56]
- Added notes/links to ion plugin. [57]
- Documented feature of initializing file. [58]
- Added output to example of creating an array withing the ion manual [59]
- Created a section about how integration tests work in Ion Shell. It also contains guides how to write integration tests for Ion Shell [24].

### 4.1.5 Configuration

- Make Linux job work in CI and fixed version in MakeFile. [60]
- Removed always overwritten test files and made it ignored in git. [61]
- Updated section about joining the redox community on matrix server. [62]

## 4.2 Issues

Resolved issues mean that they could be closed due to my contributions.

### 4.2.1 Opened and resolved Issues

- Reported problem of not working builtin documentation generation within ion manual [63]. Solved by pull request [41].
- Reported bug that white spaces between opening parentheses and the first argument causes an error [64]. Solved by pull request [64].
- Reported that ion manual is not updated by the documentation of the builtin methods [65]. Solved by pull request [66].
- Reported that the test suite of Ion Shell does not compile with intended rust version [67]. Solved by pull request [60].

### 4.2.2 Resolved Issues

The following issues were not opened by me but resolved by me via contribution.

- Feature request of new builtin method subst to allow for replacing empty arrays with an default array [21]. Resolved by pull request [19].

### 4.2.3 Opened issues

This issues were opened by me to report an problem. But they were not solved by me.

Related to redoxer

- Requested the rebuilding of the docker image for redoxer so that the Ion Shell repository can be build with a new rust version [68].

### 4.2.4 Still Opened Issues

This issues were opened by me to report an problem. But they have not been solved yet.

- Noted null-coalescing operator as a possible language feature [69]. Was brought up in the discussion in the issue [19].
- Noted down the trenary as a possible language feature [70]. Was brought up in the discussion in the issue [19].
- Reported the missing feature of handling redox os schemes in general [34].
- Proposed more information for error messages to end user. Espeacally by providing the locatoin where the error occured, ake line and colum [29].
- Reported general failing of Ci for ion shell [71].
- Reported that a job called "linux:stable" in the Ci still fails [26].

Related to redoxer.

- Reported that command "redoxer exec true" does fail on fedora [72].

# List of Figures

# List of Tables

# List of Listings

# Bibliography


[1] Redoxer Developers, "Gitlab ion shell respository," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion, (Accessed on: 2023-09-12).

[2] Ion Shell Developers, "Ion shell respository," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion, (Accessed on: 2023-09-12).

[3] ——, "Ion online manual," [Online]. Available at: https://doc.redox-os.org/ion-manual/, (Accessed on: 2023-09-12).

[4] ——, "Ion shell types," [Online]. Available at: https://doc.redox-os.org/ion-manual/variables/00-variables.html, (Accessed on: 2023-09-12).

[5] Florian Naumann, "Variable expansion," [Online]. Available at: https://doc.redox-os.org/ion-manual/expansions/01-variable.html, (Accessed on: 2023-09-18).

[6] Ion Developers, "String methods in ion shell," [Online]. Available at: https://doc.redox-os.org/ion-manual/expansions/06-stringmethods.html, (Accessed on: 2023-09-16).

[7] ——, "Arrays methods in ion shell," [Online]. Available at: https://doc.redox-os.org/ion-manual/expansions/07-arraymethods.html, (Accessed on: 2023-09-16).

[8] ——, "Ion shell if," [Online]. Available at: https://doc.redox-os.org/ion-manual/control/01-conditionals.html, (Accessed on: 2023-09-18).

[9] ——, "Ion shell loop," [Online]. Available at: https://doc.redox-os.org/ion-manual/control/01-conditionals.html, (Accessed on: 2023-09-18).

[10] ——, "Variable expansion," [Online]. Available at: https://doc.redox-os.org/ion-manual/control/01-conditionals.html, (Accessed on: 2023-09-18).

[11] Redox OS community, "Redox os book," [Online]. Available at: https://doc.redox-os.org/book/, (Accessed on: 2023-09-18).

[12] Ion shell developers, "Arrays methods in ion shell," [Online]. Available at: https://doc.redox-os.org/ion-manual/introduction.html?highlight=windows#introduction, (Local file path: src -> lib -> builtins -> helpers.rs).

[13] Ion Shell Developers, "Ion shell contributing guide lines," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/blob/master/CONTRIBUTING.md?ref_type=heads, (Accessed on: 2023-09-12).

[14] mdBook Developers, "mdbook repository," [Online]. Available at: https://github.com/rust-lang/mdBook, (Accessed on: 2023-09-18).

[15] Florian Naumann, "feat: single integration test is selectable," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1248, (Accessed on: 2023-09-12).

[16] Redox os community, "Arrays methods in ion shell," [Online]. Available at: https://matrix.to/#/#redox:matrix.org, (Local file path: src -> lib -> builtins -> helpers.rs).

[17] Redox OS community, "Creating proper pull requests in redox os," [Online]. Available at: https://doc.redox-os.org/book/ch12-04-creating-proper-pull-requests.html, (Accessed on: 2023-09-18).

[18] Rust Developers, "Testing private functions," [Online]. Available at: https://doc.rust-lang.org/book/ch11-03-test-organization.html#testing-private-functions, (Accessed on: 2023-09-12).

[19] Florian Naumann, "Implementation of subst method for arrays," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1238, (Accessed on: 2023-09-12).

[20] Nils, Eriksson matu3ba, Florian Naumann, "Issue parameter substitution on arrays," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/issues/1001, (Accessed on: 2023-09-12).

[21] Florian Naumann, "Parameter substitution on arrays," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/issues/1001, (Accessed on: 2023-09-12).

[22] ——, "Subst test code," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/blob/master/tests/subst.ion?ref_type=heads, (Accessed on: 2023-09-12).

[23] ——, "Subst expected code," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/blob/master/tests/subst.out?ref_type=heads, (Accessed on: 2023-09-12).

[24] ——, "docs: added section on how to create integration tests and how they work," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1243, (Accessed on: 2023-09-16).

[25] Timofey Prodanov, "Make aliases global," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1242, (Accessed on: 2023-09-21).

[26] Florian Naumann, "Job linux:stable fails with compile error in step make tests." [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/issues/1027, (Accessed on: 2023-09-12).

[27] ——, "feat: invalid index/range and length of sequence are provided for out of bounds error messsage to end user." [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1252, (Accessed on: 2023-09-12).

[28] ——, "feat: not found variable error now shows name of not found variable," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1253, (Accessed on: 2023-09-12).

[29] ——, "Enrich error messages information where the error occurred," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/issues/1022, (Accessed on: 2023-09-12).

[30] ——, "Build docker image of redoxer and publish new version on dockerhub," [Online]. Link to respository: https://gitlab.redox-os.org/redox-os/ion, (Local file path: src -> lib -> shell -> flow.rs).

[31] Ion Shell Developers, "Status struct within helper.rs file," [Online]. Available at: https://github.com/rust-lang/mdBook, (Local file path: src -> lib -> builtins -> helpers.rs).

[32] Florian Naumann, "fix(redox): file: prefix does not break expansion in ion shell anymore," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1239, (Accessed on: 2023-09-12).

[33] Ron Williams, "Status struct within helper.rs file," [Online]. Available at: https://gitlab.redox-os.org/redox-os/redox/-/issues/1367, (Local file path: src -> lib -> builtins -> helpers.rs).

[34] Florian Naumann, "ion cannot handle wildcards containing scheme prefixes like disk:," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/issues/1025, (Accessed on: 2023-09-12).

[35] Redoxer Developers, "Gitlab redoxer respository," [Online]. Available at: https://gitlab.redox-os.org/redox-os/redoxer, (Accessed on: 2023-09-12).

[36] Redox os community, "Arrays methods in ion shell," [Online]. Available at: https://hub.docker.com/r/redoxos/redoxer, (Local file path: src -> lib -> builtins -> helpers.rs).

[37] Florian Naumann, "Proposal: Convert makefiles and bash scripts into rust via xtask convention," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/issues/1030, (Accessed on: 2023-09-21).

[38] cargo-xtask Authors, "Respository of cargo-xtask," [Online]. Available at: https://github.com/matklad/cargo-xtask, (Accessed on: 2023-09-21).

[39] Florian Naumann, "feat: phase one, converted simple make rules into bash scripts," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1251, (Accessed on: 2023-09-21).

[40] ——, "feat: pipefail option makes pipe return error code if any non zero," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1261, (Accessed on: 2023-09-12).

[41] ——, "fix: man of history builtin is included into ion manual," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1247, (Accessed on: 2023-09-12).

[42] ——, "fix: spaces between 1. variable and opening parentheses, ignored for variable," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1250, (Accessed on: 2023-09-12).

[43] ——, "fix: in terminator removed unsafe by expect on converting bytes into utf8 str," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1261, (Accessed on: 2023-09-12).

[44] ——, "Bumped version of redox installer and fixed compile install errors," [Online]. Available at: https://gitlab.redox-os.org/redox-os/redoxer/-/merge_requests/9, (Accessed on: 2023-09-12).

[45] ——, "fix: removed redundant termination of input string," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1255, (Accessed on: 2023-09-12).

[46] ——, "fix: Removed not needed parameter as builtin map for statement lexing," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1259, (Accessed on: 2023-09-12).

[47] ——, "fix: fixed warning by replacing deprecated criterion api," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1258, (Accessed on: 2023-09-12).

[48] ——, "fix: added unit test for let in grammar," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1257, (Accessed on: 2023-09-12).

[49] ——, "feat: added test for terminator and todos about strange edge cases," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1256, (Accessed on: 2023-09-12).

[50] ——, "fix: removed not used flag field," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1244, (Accessed on: 2023-09-12).

[51] ——, "fix: removed redundant termination of input string," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1254, (Accessed on: 2023-09-12).

[52] ——, "docs: note about builtin.md is generated by make manual," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1249, (Accessed on: 2023-09-12).

[53] ——, "docs: added section on how to create integration tests and how they work," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1243, (Accessed on: 2023-09-12).

[54] ——, "docs: adding commands to open manual in browser and removed not needed destination directory," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1245, (Accessed on: 2023-09-12).

[55] ——, "fix: ignores always overwritten doc builtin file and fixes wrong formatting at generating doc builtin file," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1246, (Accessed on: 2023-09-12).

[56] ——, "fix(version): adjusted other places for minimal used rust version to 1.56.0," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1241, (Accessed on: 2023-09-12).

[57] ——, "fix(version): adjusted other places for minimal used rust version to 1.56.0," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1240, (Accessed on: 2023-09-12).

[58] ——, "docs: describes feature for executing commands at start via initrc," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1234, (Accessed on: 2023-09-12).

[59] ——, "docs: Added missing echo output for create array in manual," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1233, (Accessed on: 2023-09-12).

[60] ——, "fix: partial ci fix and fix compiler error for make tests," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1237, (Accessed on: 2023-09-12).

[61] ——, "Pull request: removed always overwritten test file for git statging," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1236, (Accessed on: 2023-09-12).

[62] ——, "Pull request: removed always overwritten test file for git statging," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1235, (Accessed on: 2023-09-12).

[63] ——, "Locally generated builtin file for ion manual does not show a man page section for all builtins of ion." [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/issues/1029, (Accessed on: 2023-09-12).

[64] ——, "Spaces between opening parentheses of method and 1. argument raises error," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/issues/1021, (Accessed on: 2023-09-12).

[65] ——, "Bug: ion manual is not updated automatically after a commit," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/issues/1019, (Accessed on: 2023-09-12).

[66] ——, "Implementation of subst method for arrays," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/merge_requests/1238, (Accessed on: 2023-09-12).

[67] ——, "Make tests does not compile with indented toolchain 1.53.0," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/issues/1018, (Accessed on: 2023-09-12).

[68] ——, "Request of publishing new version redoxer on dockerhub," [Online]. Available at: https://gitlab.redox-os.org/redox-os/redoxer/-/issues/7, (Accessed on: 2023-09-12).

[69] ——, "null-coalescing operator," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/issues/1024, (Accessed on: 2023-09-12).

[70] ——, "ternary operator for arrays/strings," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/issues/1023, (Accessed on: 2023-09-12).

[71] ——, "Ci is broken with current base image redoxos/redoxer from docker hub." [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/issues/1020, (Accessed on: 2023-09-12).

[72] ——, "Bug: Command redoxer exec true does not work on fedora," [Online]. Available at: https://gitlab.redox-os.org/redox-os/ion/-/issues/1022, (Accessed on: 2023-09-12).

# Glossary

**library** A suite of reusable code inside of a programming language for software development. i

**redoxer** Rust program to execute another rust program within an environment resembling Redox. It is used in the CI of most projects in the redox os ecosystem. It is meant to check if additions to a code base, do not prevent programs from running in Redox. Link to repository of redoxer [35]. i, 27, 29, 30

**shell** Terminal of a Linux/Unix system for entering commands. i

# Acronyms

**BDFL** Benevolent dictator for life. i

**MSRV** Minimal Supported Rust Version. i

**OS** Operating System. i