# Implementing Unix using Effect Handlers

*Douglas Torrance*


4th Year Project Report
Computer Science
School of Informatics
University of Edinburgh

2024

# Abstract

This skeleton demonstrates how to use the `infthesis` style for undergraduate dissertations in the School of Informatics. It also emphasises the page limit, and that you must not deviate from the required style. The file `skeleton.tex` generates this document and should be used as a starting point for your thesis. Replace this abstract text with a concise summary of your report.

# Research Ethics Approval

**Instructions:** *Agree with your supervisor which statement you need to include. Then delete the statement that you are not using, and the instructions in italics.*
***Either complete and include this statement:***
This project obtained approval from the Informatics Research Ethics committee.
Ethics application number: ???
Date when approval was obtained: YYYY-MM-DD
*[If the project required human participants, edit as appropriate, otherwise delete:]*
The participants' information sheet and a consent form are included in the appendix.
***Or include this statement:***
This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

# Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Douglas Torrance*)

# Acknowledgements

Any acknowledgements go here.

# Table of Contents

# Chapter 1

# Introduction

The preliminary material of your report should contain:

- The title page.
- An abstract page.
- Declaration of ethics and own work.
- Optionally an acknowledgements page.
- The table of contents.

As in this example `skeleton.tex`, the above material should be included between:

```
\begin{preliminary}
    ...
\end{preliminary}
```

This style file uses roman numeral page numbers for the preliminary material.

The main content of the dissertation, starting with the first chapter, starts with page 1. ***The main content must not go beyond page 40.***

The report then contains a bibliography and any appendices, which may go beyond page 40. The appendices are only for any supporting material that's important to go on record. However, you cannot assume markers of dissertations will read them.

You may not change the dissertation format (e.g., reduce the font size, change the margins, or reduce the line spacing from the default single spacing). Be careful if you copy-paste packages into your document preamble from elsewhere. Some LaTeX packages, such as `fullpage` or `savetrees`, change the margins of your document. Do not include them!

Over-length or incorrectly-formatted dissertations will not be accepted and you would have to modify your dissertation and resubmit. You cannot assume we will check your submission before the final deadline and if it requires resubmission after the deadline to conform to the page and style requirements you will be subject to the usual late penalties based on your final submission time.

## 1.1 Using Sections

Divide your chapters into sub-parts as appropriate.

## 1.2 Citations

# Chapter 2

# Background

## 2.1 Algebraic Effects

Algebraic effects [8] and their handlers [9] are a structured approach to managing computational side effects. They provide a way to explicitly denote computations which produce side effects and handle them non-locally.

We describe an effect by first defining its effect signature, consisting of; the operations it can perform and their input parameters and return types. This provides the interface through which we can perform a side effect. We the abstract "effect signature" and define the implementation details of handling them later. Any function which uses this effect (unless it is handling itself), must include this effect as part of its function signature.

These effects must be handled at some point using an effect handler. When the runtime encounters an effect it searches through successively outer scopes until it finds the appropriate effect handler. This allows the effect to be handled in different ways depending on the scope in which the effect was raised. The effect handler will eventually run the code which carries out the intended side effect. Effect handlers can implement interesting control flow by capturing and managing the program's continuation.

Effect handler's separate the core business logic of the program from the implementation details of handling side effects. This means side-effect handling logic is no longer scattered around the code base, improving the modularity of the code. For example, to implement logging data accesses from a database, one would usually implement logging logic in all data access functions violating the single responsibility principle. Whereas with effect handlers you can define the logging effect and handle all the logging logic in one place.

Modular code allows us to easily compose effectful operations together. When chaining effectful operations in languages like Java it is not clear where the effect is occurring and how errors are handled, making it hard to reason about how the operations will work together. This feature makes effect handler's very useful implementing side effects such as asynchronous operations, state management, logging, exception handling etc.

In the context of functional programming, effects handlers provide a simple and lightweight method of handling side effects. They are a viable alternative to the monad to handle side-effects in a functional setting. It provides a more modifiable approach compared to monads which can be rigid once they are defined. It is easier than applying monad transformers to combine side effects. It also can have better performance compared to chained monads wrapping and unwrapping values into monadic values.

## 2.2 Support for Effect Handlers

As of the time of writing this paper, effect handlers only have native support in research languages such as Koka and Effect. Some languages have concepts similar to effect handlers, for example the concept of hooks in React are similar to effect handlers in the way they allow you to handle side effects in functional components.

Features required to have a strong support for effect systems include:

- First class effect handlers
- Delimited continuations
- Optimisation for nested effects
- Effect Type System.
  - Ability to write custom effect types
  - Effect signatures greatly eases reasoning about code and indicates purity vs impurity
  - Effect type safety and inference
  - Effect polymorphism

### 2.2.1 Native Support

- Eff [1]: This language is specifically designed for algebraic effects and effect handlers. It has support for first class effect handlers, delimiting effects and abstract effects
- Koka [7]: This supports effect type inference, strongly distinguishes between pure and impure computations
- MultiCore OCaml [10]: An official extension for OCaml which provides effect typing, first class effect handlers and strong pattern matching. integrates well with handling effect cases

### 2.2.2 Library Support

Many languages have their own libraries which attempt to implement effect handlers. These implementations often come with downsides compared to native implementations. Languages which don't have first class functions which makes it difficult to compose

effects. Languages lacking an advanced type system find it difficult to create flexible and reusable handlers. Often languages' type systems don't provide type inference and ensure type safety for effects. They are also not optimised to track the multiple layers of context that an effect handler may produce. Despite this, there are some languages with effective third party libraries for using effect handlers.

- Haskell [6]: The polysemy library provides a strong implementation of effect handlers, including first class effect handlers, effect polymorphism, effect type inference. However its limited support makes it impractical to use so far.
- Scala [3]: The Cats Effect library has good performance and provides effect type polymorphism. However it doesn't provide first class effect handlers or directly support delimited continuations.

## 2.3 Syntax for Effect Handlers

### 2.3.1 Effect System

Effect Handler oriented languages have what is known as an effect system [2]. This describes the computational effects that may occur when a piece of code is executed. An effect system is typically an extension of a type system. Effect systems can be used to enforce effect safety, ensuring all effects are handled and functions only perform the side effects denotd in their effect signature

### 2.3.2 Effect Type

An effect type provides an explicit way to denote the side effects that a function performs. Each effect type can have multiple effectful operations. Each effectful operation can have parameters with value types and a valued return type. Note that operations can also produce effects and therefore have effect types.

```
effect exception
    ctl exn(error_msg : string) : a

fun safe_div (x : int, y: int) : exn int
    if y == 0 then exn("div by zero") else return x/y
```

Pure functions use the unit effect type. This shows it has no side effects.

```
fun add (x: Int, y: int) : () int {
    return x + y
}
```

## 2.4 Shallow vs Deep Handlers

Deep handlers [4] handlers can handle all the effects caused by a computation. When the continuation is captured in a handler, the captured continuation is also wrapped in

the handler. This means that deep handlers can handle effects, which themselves invoke effects.

In contrast, shallow handlers [5] only manages the first effect caused by a computation. The resumed program no longer includes the handler and the programmer needs to provide a new handler for an effect the resumption may perform. Deep and Shallow handlers can simulate each others behaviour. However it may make more sense to use one type over another in certain contexts. Most languages only support one or the other. As of 2024 deep handlers are the more popular choice amongst effect oriented languages.

## 2.5 Unix

Unix is an operating system developed by Bell Labs in the 1970s. Its aim is to create a portable, multi-user, multi-tasking system. It is responsible for managing the:

- Processes: creating, managing process communication, terminating processes, forking processes
- Scheduling
- Basic IO
- User and user environment managment

Unix is built on the “Unix Philosophy” which follows the principles of simplicity and modularity. The “everything is a file” philosophy of unix provides a consistent interface for us to interact with system resources. This will allow us to separate functionality into modules and implement them using effect handlers.

## 2.6 Evaluating Previous work

Daniel Hillerstrom has implemented a theoretical implementation of unix in his 2021 paper “Foundations for Programming and Implementing Effect Handlers” [4]. He makes analogy between operating systems and effect handlers that both interpret a series of abstract commands, in the case of the OS this is system calls, in the case of effect handlers, this is operations. The composition of effect handlers, which he views as “tiny operating systems” can provide semantics for a unix implementation. He uses deep handlers to implement; multiple user sessions, time-sharing and file IO.

This paper will use Koka to create a concrete implementation of the abstract syntax he used in his paper. Then we will implement proof by inductions to show certain properties about effect handlers.

# Chapter 3

# Conclusions

## 3.1 Final Reminder

The body of your dissertation, before the references and any appendices, *must* finish by page 40. The introduction, after preliminary material, should have started on page 1.

You may not change the dissertation format (e.g., reduce the font size, change the margins, or reduce the line spacing from the default single spacing). Be careful if you copy-paste packages into your document preamble from elsewhere. Some LaTeX packages, such as `fullpage` or `savetrees`, change the margins of your document. Do not include them!

Over-length or incorrectly-formatted dissertations will not be accepted and you would have to modify your dissertation and resubmit. You cannot assume we will check your submission before the final deadline and if it requires resubmission after the deadline to conform to the page and style requirements you will be subject to the usual late penalties based on your final submission time.

# Bibliography


[1] Andrej Bauer and Matija Pretnar. Programming with Algebraic Effects and Handlers, March 2012. arXiv:1203.1539.

[2] Andrej Bauer and Matija Pretnar. An Effect System for Algebraic Effects and Handlers. In Reiko Heckel and Stefan Milius, editors, *Algebra and Coalgebra in Computer Science*, pages 1–16, Berlin, Heidelberg, 2013. Springer.

[3] Jonathan Immanuel Brachthäuser and Philipp Schuster. Effekt: extensible algebraic effects in Scala (short paper). In *Proceedings of the 8th ACM SIGPLAN International Symposium on Scala*, pages 67–72, Vancouver BC Canada, October 2017. ACM.

[4] Daniel Hillerström. *Foundations for Programming and Implementing Effect Handlers*. PhD thesis, The University of Edinburgh, UK, 2022.

[5] Daniel Hillerström and Sam Lindley. Shallow Effect Handlers. In Sukyoung Ryu, editor, *Programming Languages and Systems*, volume 11275, pages 415–435. Springer International Publishing, Cham, 2018. Series Title: Lecture Notes in Computer Science.

[6] Oleg Kiselyov, Amr Sabry, and Cameron Swords. Extensible effects: an alternative to monad transformers. In *Proceedings of the 2013 ACM SIGPLAN symposium on Haskell*, pages 59–70, Boston Massachusetts USA, September 2013. ACM.

[7] Daan Leijen. Koka: Programming with Row Polymorphic Effect Types, June 2014. arXiv:1406.2061.

[8] Gordon D Plotkin and Matija Pretnar. Handling Algebraic Effects. *Logical Methods in Computer Science*, Volume 9, Issue 4:705, December 2013.

[9] Matija Pretnar. An Introduction to Algebraic Effects and Handlers. Invited tutorial paper. *Electronic Notes in Theoretical Computer Science*, 319:19–35, December 2015.

[10] KC Sivaramakrishnan, Stephen Dolan, Leo White, Tom Kelly, Sadiq Jaffer, and Anil Madhavapeddy. Retrofitting effect handlers onto OCaml. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, PLDI 2021, pages 206–221, New York, NY, USA, June 2021. Association for Computing Machinery.

# Appendix A

# First appendix

## A.1 First section

Any appendices, including any required ethics information, should be included after the references.

Markers do not have to consider appendices. Make sure that your contributions are made clear in the main body of the dissertation (within the page limit).

# Appendix B

# Participants' information sheet

If you had human participants, include key information that they were given in an appendix, and point to it from the ethics declaration.

# Appendix C

# Participants' consent form

If you had human participants, include information about how consent was gathered in an appendix, and point to it from the ethics declaration. This information is often a copy of a consent form.