

School of Computer Science and Software Engineering
Monash University

Bachelor of Computer Science Honours (1608), Clayton Campus

Research Proposal — Semester 1, 2002

Structured process input/output

Reformulating the UNIX command line environment to generate and process XML documents

Cameron McCormack (12793086)

Supervisor: Prof. John Hurst

Table of Contents

1. Introduction	3
2. Research Context	3
3. Research Plan and Methods	4
3.1. Methodology	4
3.2. Proposed Thesis Section Headings	5
3.3. Timetable	5
3.4. Special Facilities	6
4. Bibliography	6

1. Introduction

One of the strengths of the UNIX operating system (Ritchie, Thompson 1974) is its philosophy of having many small, simple tools which perform specific tasks well. Users can compose commands to perform more complex computations by combining these small tools at the command line, by way of a pipeline. This is especially true of text processing, where the tools can act as "filters", consuming some text from its input, and producing a modified version of the text on its output.

Back in the early days of UNIX, text processing typically consisted of manipulating flat text files. This data file format is simply a collection of ASCII-based records, separated by newline characters. The standard text processing tools, such as `grep`, `awk` and `sed`, all focus on this flat text file format.

There are, however, problems with dealing solely with flat text. Firstly, not all data easily transform into this format; kludges must be made to fit the data into records. As an example, the hierarchical structure of a directory tree would benefit from a textual representation that could accommodate the nesting of directory entries in other directory entries.

Secondly, some programs ignore the inherent record-based nature of their output and instead format it to be friendly for the user. To process this requires the use of text filtering programs solely to extract the relevant information from the formatted output. A prime example of this would be UNIX's `ls -l` command. The output from this command contains a date field whose format can change according to its difference from the current date. Such conditional formatting makes it difficult to consistently parse the information.

This project aims to look at the use of structured text for standard program input and output, in particular the Extensible Markup Language (XML), as a means of solving these problems. Following on from this, an environment in which to run these programs will also be devised.

2. Research Context

The idea of shifting the focus of UNIX process I/O from flat text to a more structured format has not been investigated greatly. Kernighan & Plauger (1976) introduced the idea of the "software tool" — the small program which performed its task, and only its task. Descriptions of the standard UNIX text processing tools and combining these using the shell are presented (Kernighan, Pike 1984). However, all of these tools are designed for flat text processing.

The beginnings of structured, marked up document processing lie in the Standard Generalized Markup Language (SGML) (ISO 8879:1986). SGML found itself a following in the publishing industry, as well as a killer application in the form of the Hypertext Markup Language (HTML) (Berners-Lee, Connolly 1995). It wasn't until the formulation of a simpler, extended version of SGML, the Extensible Markup Language (XML), that a standard, structured document format was used for general computer interchange of information (Bray, Sperberg-McQueen 1996).

Recently, XML has been proposed as the data format for many tasks, such as Remote Procedure Call (RPC) messages (Box, et al 2000), however XML hasn't yet been seriously considered for general process I/O.

3. Research Plan and Methods

3.1. Methodology

For the XML processing environment to be successful, there must be three main components: tools that generate XML documents, tools which process and filter XML documents, and an interface to compose these tools. The project will look at each of these three components in turn.

The first stage of the project will involve identifying the UNIX tools which are text producers, and adapting these tools to produce XML output instead of flat text. Two important text producing tools will be studied: `ls` and `ps`. The techniques for converting these programs to be XML generating will be discussed and then applied to the other text producing tools.

The second stage will look at the range of standard text filtering tools available to the UNIX user. Two commonly used filters, `grep` and `sed` will be expounded upon. Again, versions of these tools which use XML documents as their input and output will be developed.

Lastly, a shell to handle the composition, execution and output presentation of these XML tools will be devised. The emphasis for this shell will be on connecting the XML tools together using pipes, as the regular UNIX shell does, and on providing a way of automatically converting the output of these tools to a form suitable for display to the user. The composition of process using pipes will not be substantially different from the Bourne shell, however issues of document metadata will be considered. Process output document formatting will be handled using XML Stylesheet Transformations (XSLT) (Clark 1999).

The formulation of the overall XML tool environment will hopefully provide benefits to the user, mostly in the form of programs generating output from which it is easy to extract relevant information.

3.2. Proposed Thesis Section Headings

1. Introduction
2. The state of UNIX tools
3. Text producing tools
4. Text processing and filtering tools
5. A user interface for the XML tools
6. Conclusions and future work
7. Bibliography
8. Appendix A: Software developed

3.3. Timetable

Date	Task
March 25	Begin reading literature
April 1	Begin coding for stage 1
April 15	Start writing research proposal
April 29	Show draft proposal to supervisor
May 2	Submit research proposal
May 13	Prepare for interim presentation
May 20	Begin literature review
May 23	Interim presentation
May 27	Begin coding for stage 2
June 13	Submit literature review draft
July 8	Begin coding for stage 3
August 1	Submit literature review
August 22	Finalise coding, begin thesis
September 12	Submit thesis draft
October 10	Prepare final presentation
October 17	Final presentation
November 1	Submit research log book
November 5	Submit final thesis
November 11	Finalise project web site

3.4. Special Facilities

No facilities besides those made available to all honours students at the School of Computer Science and Software Engineering at Monash University will be necessary for the completion of the research as described in this proposal.

4. Bibliography

International Organization for Standardization (1986), *Information processing — Text and office systems — Standard Generalized Markup Language (SGML)*, ISO 8879:1986, Geneva.

Berners-Lee, T. & Connolly, D. (1995, Nov.), "Hypertext Markup Language — 2.0," The Internet Engineering Taskforce [online], Available: <http://www.ietf.org/rfc/rfc1866.txt> [Accessed 2 May 2002].

Box, D., et al (2000, May 8), "Simple Object Access Protocol (SOAP) 1.1," The World Wide Web Consortium [online], Available: <http://www.w3.org/TR/2000/NOTE-SOAP-20000508> [Accessed 2 May 2002].

Bray, T. & Sperberg-McQueen, C.M. (1996, Nov. 14), "Extensible Markup Language (XML)," The World Wide Web Consortium [online], Available: <http://www.w3.org/TR/WD-xml-961114.html> [Accessed 2 May 2002].

Clark, J. (1999, Nov. 16), "XSL Transformations (XSLT)," The World Wide Web Consortium [online], Available: <http://www.w3.org/TR/1999/REC-xslt-19991116> [Accessed 2 May 2002].

Kernighan, B.W. & Pike, R. (1984), *The UNIX Programming Environment*, Prentice Hall, Inc., Englewood Cliffs, New Jersey.

Kernighan, B.W. & Plauger, P.J. (1976), *Software Tools*, Addison-Wesley, Reading, Massachusetts.

Ritchie, D.M. & Thompson, K. (1974), "The UNIX time-sharing system," *Communications of the ACM*, vol. 17, no. 7, pp. 365—375.