

Structured Process Input/Output

Cameron McCormack

School of Computer Science and Software Engineering

What's the project about?

- Moving away UNIX process I/O from flat text (line-based records) to structured text (XML documents)

The UNIX Command Line Environment

The UNIX Command Line Environment

- Why is it popular?
 - ◆ Idea of the "software tool" (Kernighan 1976)
 - ◆ Many simple, specific programs
 - ◆ Composition of these simple programs to make complex ones

The UNIX Command Line Environment

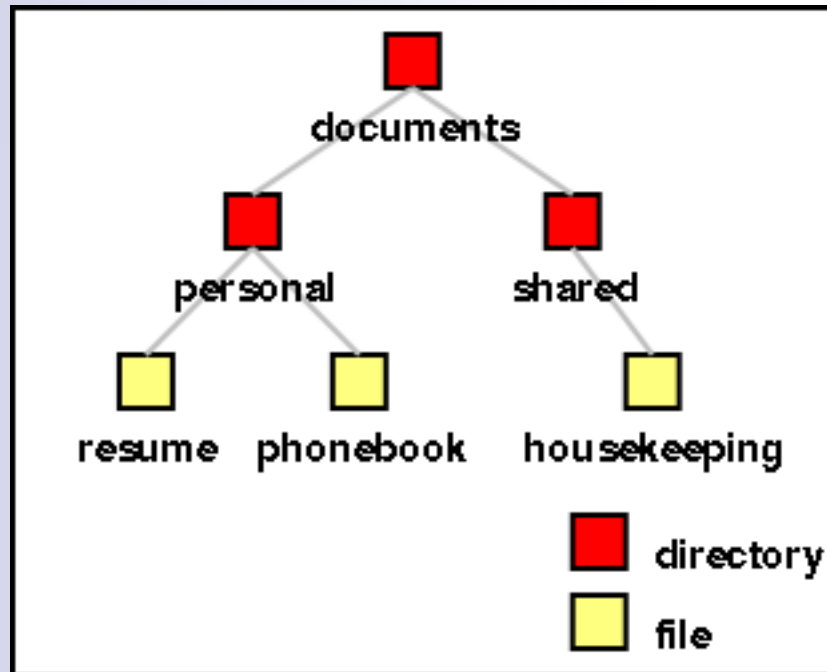
- Why is it popular?
 - ◆ Idea of the "software tool" (Kernighan 1976)
 - ◆ Many simple, specific programs
 - ◆ Composition of these simple programs to make complex ones
- Focus is on flat text processing
 - ◆ Line-based records
 - ◆ Fields separated by whitespace
 - ◆ Is this a problem?

A problem with flat text

- Not all data conform to this record/field model

A problem with flat text

- Not all data conform to this record/field model
- An example
 - ◆ A hierarchical directory listing



An example

- Output of "ls -R"

```
.:  
documents
```

```
./documents:  
personal  
shared
```

```
./documents/personal:  
phonebook  
resume
```

```
./documents/shared:  
housekeeping
```

An example

- Output of "ls -R"

```
.:  
documents
```

```
./documents:  
personal  
shared
```

```
./documents/personal:  
phonebook  
resume
```

```
./documents/shared:  
housekeeping
```

- The hierarchical information doesn't fit into records/fields

An example

- Output of "ls -R"

```
.:  
documents
```

```
./documents:  
personal  
shared
```

```
./documents/personal:  
phonebook  
resume
```

```
./documents/shared:  
housekeeping
```

- The hierarchical information doesn't fit into records/fields
- Using this output, how do you find all files two directories deep?

Another problem

- Many programs format output for human reading
- More difficult for programs to parse

Another example

- Output of ls long format

```
$ ls -l
total 0
-rw-r--r--    1 cameron  cameron    253 May 22 23:40 phonebook
-rw-r--r--    1 cameron  cameron    763 Sep  5  2001 resume
```

Another example

- Output of ls long format

```
$ ls -l
total 0
-rw-r--r--    1 cameron  cameron    253 May 22 23:40 phonebook
-rw-r--r--    1 cameron  cameron    763 Sep  5  2001 resume
```

- How do you extract the modified time?

- ◆ No clear field delimiter
- ◆ Normally use "cut"
- ◆ But this needs knowledge of output format
- ◆ Modified time format can also change

Another example

- Output of ls long format

```
$ ls -l
total 0
-rw-r--r--    1 cameron  cameron    253 May 22 23:40 phonebook
-rw-r--r--    1 cameron  cameron    763 Sep  5  2001 resume
```

- How do you extract the modified time?

- ◆ No clear field delimiter
- ◆ Normally use "cut"
- ◆ But this needs knowledge of output format
- ◆ Modified time format can also change

- Need to separate information from presentation

Another example

- Output of ls long format

```
$ ls -l
total 0
-rw-r--r--    1 cameron  cameron    253 May 22 23:40 phonebook
-rw-r--r--    1 cameron  cameron    763 Sep  5  2001 resume
```

- How do you extract the modified time?

- ◆ No clear field delimiter
- ◆ Normally use "cut"
- ◆ But this needs knowledge of output format
- ◆ Modified time format can also change

- Need to separate information from presentation

- So, what can we do?

The solution

- Incorporate a well-defined structure into process' input/output

The solution

- Incorporate a well-defined structure into process' input/output
- XML as the data format
 - ◆ Standardised
 - ◆ Parsers already exist
 - ◆ Still human readable
 - ◆ Hierarchical
 - ◆ Added benefit: Unicode

The updated ls

- The output generated by our new ls program:

```
<ls-output>
  <file name="documents" path="." uid="1000" gid="1000"
    size="4096" type="directory" mode="0755"
    mtime="1022074804">
    <file name="personal" uid="1000" gid="1000" size="4096"
      type="directory" mode="0755" mtime="1022074845">
      <file name="phonebook" uid="1000" gid="1000" size="253"
        type="regular" mode="0644" mtime="1022074845"/>
      <file name="resume" uid="1000" gid="1000" size="763"
        type="regular" mode="0644" mtime="1022078771"/>
    </file>
    <file name="shared" uid="1000" gid="1000" size="4096"
      type="directory" mode="0755" mtime="1022074911">
      <file name="housekeeping" uid="1000" gid="1000" size="0"
        type="regular" mode="0644" mtime="1022074911"/>
    </file>
  </file>
</ls-output>
```

The updated ls

- The output generated by our new ls program:

```
<ls-output>
  <file name="documents" path="." uid="1000" gid="1000"
    size="4096" type="directory" mode="0755"
    mtime="1022074804">
    <file name="personal" uid="1000" gid="1000" size="4096"
      type="directory" mode="0755" mtime="1022074845">
      <file name="phonebook" uid="1000" gid="1000" size="253"
        type="regular" mode="0644" mtime="1022074845"/>
      <file name="resume" uid="1000" gid="1000" size="763"
        type="regular" mode="0644" mtime="1022078771"/>
    </file>
    <file name="shared" uid="1000" gid="1000" size="4096"
      type="directory" mode="0755" mtime="1022074911">
      <file name="housekeeping" uid="1000" gid="1000" size="0"
        type="regular" mode="0644" mtime="1022074911"/>
    </file>
  </file>
</ls-output>
```

- Now how easy is it to
 - ◆ Find files two directories deep
 - ◆ Extract the last modified time

What else do we need?

- Without more XML aware programs, it is still difficult to extract information from this output

What else do we need?

- Without more XML aware programs, it is still difficult to extract information from this output
- We need some filters
 - ◆ An equivalent for cut, grep etc.
 - ◆ Some of these filters can utilise XPath

What else do we need?

- Without more XML aware programs, it is still difficult to extract information from this output
- We need some filters
 - ◆ An equivalent for cut, grep etc.
 - ◆ Some of these filters can utilise XPath
- We can now say

```
$ ls -R | select '/ls-output/file/file/file'  
<ls-output>  
  <file name="phonebook" uid="1000" gid="1000" size="253"  
    type="regular" mode="0644" mtime="1022074845"/>  
  <file name="resume" uid="1000" gid="1000" size="763"  
    type="regular" mode="0644" mtime="1022078771"/>  
  <file name="housekeeping" uid="1000" gid="1000" size="0"  
    type="regular" mode="0644" mtime="1022074911"/>  
</ls-output>
```

What about output for human consumption?

- This XML output is not suitable for presenting to the user

What about output for human consumption?

- This XML output is not suitable for presenting to the user
- Need to transform it to some format for the terminal
 - ◆ Can use XSLT to do the transformation
 - ◆ But it is unwieldy to manually transform every command's output

What about output for human consumption?

- This XML output is not suitable for presenting to the user
- Need to transform it to some format for the terminal
 - ◆ Can use XSLT to do the transformation
 - ◆ But it is unwieldy to manually transform every command's output
- Transformation must happen automatically
 - ◆ We need support from the shell

The new shell

- Handles composition of programs just like Bourne shell

The new shell

- Handles composition of programs just like Bourne shell
- But also detects output type and transforms it appropriately

```
$ ls -R | select '/ls-output/file/file'  
-rw-r--r--      1 cameron  cameron      253 May 22 23:40 phonebook  
-rw-r--r--      1 cameron  cameron      763 May 23 00:46 resume  
-rw-r--r--      1 cameron  cameron        0 May 22 23:41 housekeepin
```

The new shell

- Handles composition of programs just like Bourne shell
- But also detects output type and transforms it appropriately

```
$ ls -R | select '/ls-output/file/file'  
-rw-r--r--      1 cameron  cameron    253 May 22 23:40 phonebook  
-rw-r--r--      1 cameron  cameron    763 May 23 00:46 resume  
-rw-r--r--      1 cameron  cameron     0 May 22 23:41 housekeepin
```

- Issues of metadata

My project

- Split into three stages

My project

- Split into three stages
- Stage 1
 - ◆ Identify standard UNIX text generating programs
 - ◆ ls, ps, netstat, etc.
 - ◆ Modify or wrap them to generate XML

My project

- Split into three stages
- Stage 1
 - ◆ Identify standard UNIX text generating programs
 - ◆ ls, ps, netstat, etc.
 - ◆ Modify or wrap them to generate XML
- Stage 2
 - ◆ Identify standard UNIX text filtering programs
 - ◆ grep, cut, awk, etc.
 - ◆ Modify them to filter XML documents

My project

- Split into three stages
- Stage 1
 - ◆ Identify standard UNIX text generating programs
 - ◆ ls, ps, netstat, etc.
 - ◆ Modify or wrap them to generate XML
- Stage 2
 - ◆ Identify standard UNIX text filtering programs
 - ◆ grep, cut, awk, etc.
 - ◆ Modify them to filter XML documents
- Stage 3
 - ◆ Write (simplistic) shell similar to Bourne shell
 - ◆ The shell will handle presentation issues
 - ◆ Will transform output for human reading on the terminal

Project status

- At this stage, implemented two text generating programs from Stage 1
 - ◆ ls and ps
- Just started looking at Stage 2

Concluding remarks

- For more information on my project, see <http://www.csse.monash.edu.au/~clm/home/uni/honours/>.
- Thanks for listening!