

Structured Process Input/Output

Cameron McCormack
Supervisor: John Hurst

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

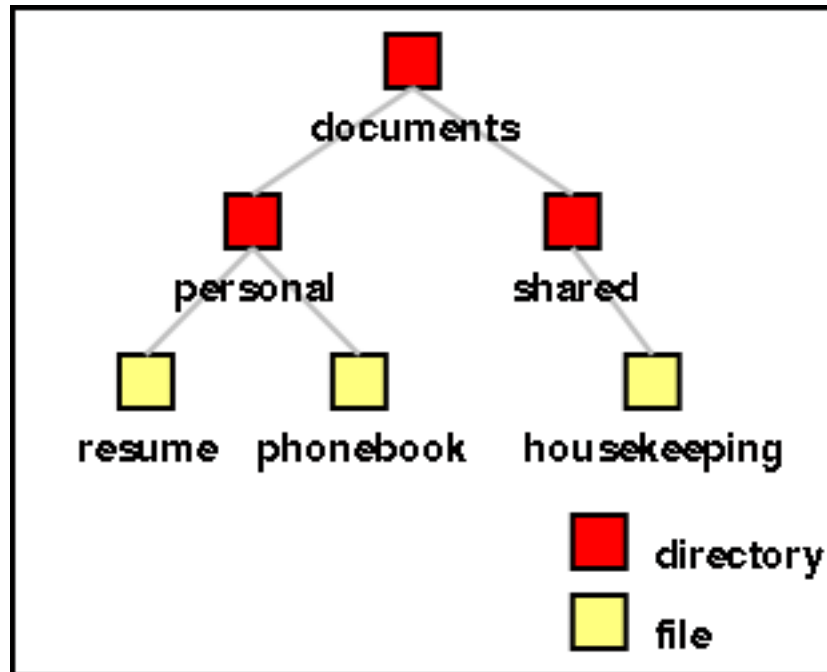
- Why is it popular?
 - ◆ Idea of the "software tool" (Kernighan 1976)
 - ◆ Many simple, specific programs
 - ◆ Composition of these simple programs to make complex ones
- Focus is on flat text processing
 - ◆ Line-based records
 - ◆ Fields separated by whitespace
 - ◆ Is this a problem?

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 Oct 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 Oct 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 Oct 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 Oct 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 new model - programs

- Programs divided into three categories
 - ◆ Data generating programs
 - ◆ Data filtering programs
 - ◆ User interface to the environment (the shell)

The new model - data format

- Programs take an XML document on standard input and standard output
- Standard error remains plain text

The new model - data format

- Programs take an XML document on standard input and standard output
- Standard error remains plain text
- Most programs will generate documents of the form:

```
<records xmlns="urn:ns:clm.xml-unix.XXXX">  
  <record attributel="..." attributen="...">  
    <more-data/>  
  </record>  
  
  <record attributel="..." attributen="...">  
    <more-data/>  
  </record>  
</records>
```

Data generating programs

- Programs such as ls, ps, df
- These programs take no input
- Not responsible for formatting the output

Data generating programs

- Programs such as ls, ps, df
- These programs take no input
- Not responsible for formatting the output
- They do have formatting options, however
- Formatting preferences recorded, but not acted upon until later

```
<addressbook xmlns="urn:ns:clm.xml-unix.address"
             hidework="true">
  <entry id="1">
    <name>
      <family>Smythe</family>
      <given>Jon</given>
    </name>
    <home>
      <phone>9555 1234</phone>
    </home>
    <work>
      <phone>9600 4321</phone>
    </work>
  </entry>
  ..
</addressbook>
```

The updated ls

- The output generated by our new ls program:

```
<files xmlns="urn:ns:clm.xml-unix.ls">
  <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>
</files>
```

The updated ls

- The output generated by our new ls program:

```
<files xmlns="urn:ns:clm.xml-unix.ls">
  <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>
</files>
```

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

Data filtering programs

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

Data filtering programs

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

Data filtering programs

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

```
$ ls -R | select '/ls-output/file/file/file'  
<files xmlns="urn:ns:clm.xml-unix.ls">  
  <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"/>  
</files>
```

User interface

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

User interface

- 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

User interface

- 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
```

Automatic transformation

- The shell inspects the output of the command
- It looks at the namespace of the document element

```
<files xmlns="urn:ns:clm.xml-unix.ls">
```

```
  ..  
</files>
```

- It checks /etc/transforms.xml to determine which XSLT stylesheet to transform the output with
- It runs the XSLT transformer, the output going to the terminal

Prevent automatic transformation

- Allow the user to view the XML source, if they wish
- Helpful for determining how to construct a command pipeline

Prevent automatic transformation

- Allow the user to view the XML source, if they wish
- Helpful for determining how to construct a command pipeline
- The ^ character tells the shell to keep the markup

```
$ df ^
<filesystems xmlns="urn:ns:clm.xml-unix.df">
  <filesystem dev="/dev/hde2" blocks="29249536" free="2113536"
    use="93" mountpoint="/" />
  <filesystem dev="proc" blocks="0" free="0" use="0"
    mountpoint="/proc" />
  <filesystem dev="/dev/hdf3" blocks="28409856" free="1589248"
    use="95" mountpoint="/storage" />
  <filesystem dev="usb" blocks="0" free="0" use="0"
    mountpoint="/proc/bus/usb" />
</filesystems>
```

What we can achieve with this

- We can now work with the underlying information without trying to get around the formatting of one program
- Allows the user to construct commands more intuitively

Limitations of the model

- Not everything is suited to XML!
 - ◆ Non XML documents must be supported as well

Limitations of the model

- Not everything is suited to XML!
 - ◆ Non XML documents must be supported as well
- All documents must be well formed
 - ◆ Does it make sense to run a filter on part of a document tree?
 - ◆ An example: XInclude

Demonstration

Significant outcomes

Significant outcomes

- Positive
 - ◆ Commands to extract data are easier to produce
 - ◆ Programs are simplified, not having to worry about presentation
 - ◆ Named fields are important

Significant outcomes

- Positive
 - ◆ Commands to extract data are easier to produce
 - ◆ Programs are simplified, not having to worry about presentation
 - ◆ Named fields are important
- Negative
 - ◆ Users must think about the problem at a different level
 - ◆ Must work from the source, not just from what's on the screen

Conclusions drawn

- Separation of form and content is definitely an advantage
- Programs interoperate better with a standard file format

Conclusions drawn

- Separation of form and content is definitely an advantage
- Programs interoperate better with a standard file format
- Simplicity is lost
- Will be difficult to get people to change

Future work

- A UNIX-like operating system for this environment to be maximally useful in

Future work

- A UNIX-like operating system for this environment to be maximally useful in
- Extending the environment to access network objects (XML over HTTP)

Thanks for listening!